PR^2: PREDICTIVE ROUTING REPLAY FOR MOE-BASED LLM REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture of Experts (MoE) Large Language Models (LLMs) achieve strong performance at scale. However, reinforcement learning (RL) on MoE-based LLMs often suffers from training instability. A root cause is *router drift*, i.e., expert activations can change drastically across model updates and differ between disaggregated rollout and training phases, causing large rollout–training mismatch and unstable importance sampling weights in PPO-style RL algorithms. Routing replay mitigates this issue by freezing the replay route within each reasoning trajectory, but it ignores how the router evolves under off-policy updates and thus causes *router staleness*. We introduce **Predictive Routing Replay (PR^2)**, which augments each router with a lightweight evolution predictor that learns to anticipate short-horizon router evolution. During route recording, the predictor selects expert sets that anticipate router evolution. During training, it replays these sets with the current model. We bound recorded-route sensitivity through predictor fit and update drift, derive an exact KL threshold for top-k agreement, and show how local support certificates yield matching forward computations and fixed-batch policy gradients. Across three MoE models and reuse factors 2, 4 and 8, PR^2 achieves the highest average accuracy in all nine comparisons, improving average math accuracy over routing replay by 8.16 to 13.00 percentage points on Qwen3-30B-A3B-Base.

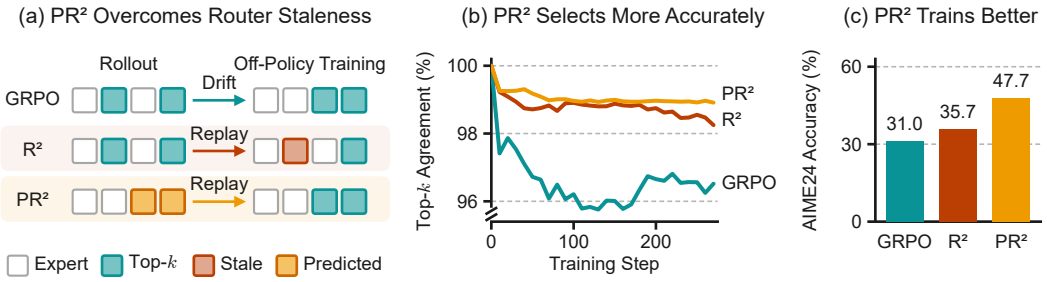


Figure 1: **PR^2 overcomes router staleness.** (a) One token with four experts and $k = 2$: the router drifts between rollout and off-policy training (GRPO); routing replay freezes the recorded route, so part of it goes stale (R^2); PR^2 records a predicted route that matches the router after the updates. (b) Top-k agreement between the cached route and the current router during training. (c) PR^2 reaches the highest AIME24 accuracy (Avg@32). Panels (b) and (c) use Qwen3-30B-A3B-Base under off-2.

1 INTRODUCTION

Large language models (LLMs) have demonstrated strong reasoning capabilities when scaled through increased model parameters and training compute (OpenAI, 2024; DeepSeek-AI, 2026; Gemini Team, 2025). Mixture of Experts (MoE) models have recently emerged as a promising LLM architecture, enabling training compute to scale sublinearly with model size (Jiang et al., 2024; Zhu et al., 2024; Liu et al., 2024a; Yang et al., 2025; Agarwal et al., 2025; Kimi Team et al., 2025; GLM Team, 2026; DeepSeek-AI, 2026). This efficiency is achieved by activating a subset of expert networks within the large model, where the activated parameters typically account for less than 10% of the total model parameters (Liu et al., 2024a; Yang et al., 2025; GLM Team, 2026). This conditional computation paradigm has enabled models with trillions of parameters with manageable inference costs, and has become a core design choice in large-scale LLMs.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Beyond pretraining, reinforcement learning (RL) has emerged as a central mechanism for optimizing pretrained LLMs to improve reasoning and agentic capabilities, as well as alignment with human preferences (Ouyang et al., 2022; Rafailov et al., 2023; Shao et al., 2024; Liu et al., 2025b; Gao et al., 2025b). Algorithms such as Proximal Policy Optimization (PPO) (Schulman et al., 2017) and its variants (Shao et al., 2024; Guo et al., 2025; Yu et al., 2025) are now widely used to improve reasoning, long-horizon decision making, and tool use tasks (Chen et al., 2022; Yao et al., 2023). However, applying RL to MoE-based LLMs introduces a unique and underexplored set of stability challenges that do not arise in dense models (Yao et al., 2025).

A key difficulty for conducting stable RL on MoE-based LLMs stems from the presence of learned routers that dynamically assign tokens to experts (Yao et al., 2025; Zheng et al., 2025a; Kim et al., 2025). When policy updates reuse trajectories generated by a stale snapshot, MoE models expose this off-policy gap as *router drift*, where the same token may be routed to different experts by the old snapshot and the current training policy. Such routing changes alter the computation path behind PPO-style importance ratios and can amplify their variance, thereby destabilizing policy optimization (Schulman et al., 2017; Shao et al., 2024; Yu et al., 2025).

To mitigate routing mismatch, routing replay records expert routes before training updates and reuses them during gradient evaluation; the old-snapshot variant records routes from $\pi_{\theta_{\text{old}}}$ (Zheng et al., 2024). The resulting route consistency fixes the expert sets to a past router state. By fixing the cached routes, it prevents gradients from reaching experts that would become active after subsequent policy updates. Over time, this induces *router staleness*, as newly activated experts under the updated routing policy are excluded from gradient updates. Theoretically, this yields stale gradient estimation under replayed routes, where gradients are evaluated under a fixed replay route whose distribution deviates from the evolving current training policy.

In this work, we revisit routing replay from a theoretical perspective. We formalize routing replay as replacing the current routing distribution with a degenerate replay measure and derive a bound on router staleness, showing that replay staleness controls route-induced gradient deviation in fixed-route PPO gradients. This view reveals:

> *Stable importance estimation favors frozen routing, while effective learning requires routing distributions that track policy evolution.*

Motivated by this perspective, we propose **Predictive Routing Replay (PR^2)**, which augments routing replay with a route prediction scheme (Figure 2). PR^2 augments the router in each MoE layer with a lightweight *evolution predictor* that anticipates *router drift*. During the old-logprob recording pass, the predictor outputs a learned logit bias to adjust the routing distribution. We then perform top-k routing under the biased logits to obtain a predicted expert index, which is cached alongside trajectories for subsequent updates. During the training phase, we replay the cached top-k expert indices while disabling the predictor, keeping the replay route fixed to stabilize PPO-style importance ratios yet allowing gradient flow to experts that are likely to become active after policy updates. We train the predictor on cached features with a KL fitting objective and a dedicated learning-rate multiplier. Our analysis connects fitting error and predictor updates to the recorded route, and the learned predictor selects routes for the next batch.

PR^2 can be integrated into existing MoE RL frameworks, e.g., VeRL (Sheng et al., 2025), Slime (Zhu et al., 2025), Miles (Team, 2026) and TRL (von Werra et al., 2020), without modifying the policy optimization objective. Empirically, PR^2 substantially reduces routing mismatch, stabilizes RL training, and improves performance over strong baselines (Guo et al., 2025; Yu et al., 2025; Zheng et al., 2025b) across reasoning tasks (Figure 1). On AIME24 (Zhang & Math-AI, 2025), GRPO with PR^2 achieves 40.31% accuracy on Qwen3-30B-A3B-Base (Yang et al., 2025), improving over routing replay and GSPO by 12.29 and 9.38 percentage points under off-8, respectively.

Our main contributions are summarized below.

- We formalize *router staleness* as a key source of degradation in routing replay for MoE-based LLM reinforcement learning, and derive a divergence-based bound showing that replay staleness controls route-induced gradient deviation in fixed-route PPO gradients.
- We propose **Predictive Routing Replay (PR^2)**. We connect predictor fit and update drift to replay sensitivity, bound top-k support regret, and derive an exact KL radius for support agreement with a complete-computation consistency result.

- Across three MoE models and three reuse factors, PR^2 achieves the highest average accuracy in all nine comparisons, with gains of up to 13.00 percentage points over routing replay. Routing analyses and repeated-run studies further characterize its behavior.

2 Related Work

Mixture of Experts. The Mixture of Experts (MoE) layer scales model capacity through conditional computation, where a learned router selects a subset of expert networks per token to reduce FLOPs while retaining a large number of parameters (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022; Zhu et al., 2024; Jiang et al., 2024; Yang et al., 2025; Kimi Team et al., 2025). A large body of work studies the routing strategy (Lewis et al., 2021; Huang et al., 2024), load balancing (Shazeer et al., 2017; Fedus et al., 2022; Liu et al., 2024a), and gradient estimation (Kool et al., 2021; Liu et al., 2023; 2024b) in MoE training. Analysis also highlights that routing decisions can be fragile (Dai et al., 2022; Zoph et al., 2022), which leads to unstable gradient flow and optimization during training (Kim et al., 2025). This fragility becomes acute when systems separate rollout from training (Yao et al., 2025), which is the setting we address.

Reinforcement Learning in LLMs. Reinforcement learning (RL) is widely used to align LLMs for improved reasoning and instruction following, with PPO-style algorithms and their variants serving as widely used optimization methods (Ouyang et al., 2022; Shao et al., 2024; Guo et al., 2025; Yu et al., 2025; Zhao et al., 2026). Large-scale LLM RL is often implemented in disaggregated pipelines, where rollouts and training are lagged, thereby introducing off-policy effects (Mnih et al., 2016; Espeholt et al., 2018). For MoE-based LLMs, this mismatch is exacerbated by *router drift*, where the same token can be assigned to different experts during rollout and training forward passes. This routing mismatch can amplify importance-ratio variance and destabilize policy optimization, harming RL performance (Yao et al., 2025).

Stabilizing RL in MoE. To overcome *router drift*, routing replay caches expert indices and reuses them during training to ensure routing consistency; the old-snapshot variant (Zheng et al., 2024) records routes from the old policy snapshot. Another variant, rollout routing replay (Ma et al., 2025), targets the rollout-engine setting. Beyond routing replay, GSPO (Zheng et al., 2025b) defines the importance ratio based on sequence likelihood and performs sequence-level clipping, aggregating token likelihood changes before clipping. Other methods improve stability by modifying the clipping behavior (Gao et al., 2025a) or directly adjusting importance ratios to reduce sensitivity to off-policy errors (Zhang et al., 2025). These remedies either replay an observed route or reweight and clip the ratio computed on it. PR^2 changes which route is replayed: it replays the route the trained router is moving toward, and it composes with objective-level methods such as GSPO (Appendix G).

3 Preliminaries

3.1 Off-Policy Reinforcement Learning

Let μ be the behavior policy that generates a completion $y = (y_1, \ldots, y_T)$ for prompt x, and let π_θ be the current policy. With $s_t = (x, y_{<t})$, the token likelihood ratio and trajectory weight are

$$w_t(\theta) = \frac{\pi_\theta(y_t \mid s_t)}{\mu(y_t \mid s_t)}, \qquad w_\theta(x, y) = \prod_{t=1}^{T} w_t(\theta). \tag{1}$$

For common prompt sampling and positive behavior probabilities, the reward objective is $J(\theta) = \mathbb{E}_{x,y\sim\mu}[w_\theta(x, y)R(x, y)]$. PPO-style methods optimize clipped likelihood-ratio surrogates with fixed advantages (Schulman et al., 2017; Shao et al., 2024; Guo et al., 2025). In a disaggregated pipeline, the training engine records an old likelihood ν_t for each sampled token. The snapshot ratio $r_t = \pi_\theta(y_t \mid s_t)/\nu_t$ and the behavior correction $c_t = \nu_t/\mu(y_t \mid s_t)$ satisfy $w_t = c_t r_t$. Appendix A specifies the objectives, and Appendix C gives the replay-policy factorization.

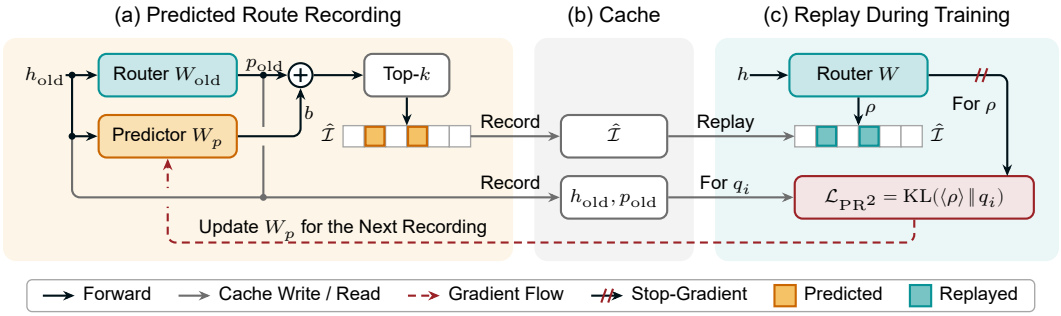


Figure 2: **Overview of Predictive Routing Replay (PR^2).** One RL iteration, shown for one MoE layer and one token. (a) When routes are recorded, the evolution predictor adjusts the old router's logits, and top-k selection gives the predicted route. (b) The cache stores the predicted route and the router features. (c) Every training update replays the cached route without a fresh top-k, and the predictor is trained on the cached features for the next round (Section 4).

3.2 Reinforcement Learning on Mixture of Experts

An MoE model has L routed layers and N experts per layer. At token t and layer l, the router selects a set $\mathcal{I}_t^{(l)}$ of k experts. Write $\mathcal{R}_t = \{\mathcal{I}_t^{(l)}\}_{l=1}^{L}$ and $\mathcal{R}_{\le t}$ for the routing events needed to evaluate the prefix, including its key–value states. A distribution $\rho^{\pi}_{\theta,\le t}$ over these routes gives

$$\pi_\theta(y_t \mid s_t) = \mathbb{E}_{\mathcal{R}_{\le t}\sim\rho^{\pi}_{\theta,\le t}}[\pi_\theta(y_t \mid s_t, \mathcal{R}_{\le t})]. \tag{2}$$

Deterministic top-k routing induces a point mass on the selected route.

Router Drift and Replay. As the policy changes, the same prefix can activate different experts. This router drift changes the computation paths used to evaluate likelihood ratios and can amplify their variance (Zheng et al., 2025a; Yao et al., 2025). Routing replay (Zheng et al., 2024) caches expert sets $\tilde{\mathcal{R}}_{\le t}$ and uses the conditional policy

$$\pi_\theta^{\tilde{\mathcal{R}}}(y_t \mid s_t) := \pi_\theta(y_t \mid s_t, \tilde{\mathcal{R}}_{\le t}) \tag{3}$$

in subsequent training passes. Replay keeps the selected indices fixed while expert parameters and combination weights evolve, so the replayed route falls behind the drifting router.

3.3 Router Staleness

We call the divergence between the current router and the recorded routing preference *router staleness*. For route distributions P_t and Q_t, define $\mathcal{S}_t = D_{\text{TV}}(P_t, Q_t)$. Let $\psi_t(\mathcal{R}, \theta)$ be the gradient of a token loss with route $\mathcal{R}$ fixed, and let $g_t(P_t, \theta) = \mathbb{E}_{\mathcal{R}\sim P_t}[\psi_t(\mathcal{R}, \theta)]$. At common parameters, prefixes, advantages and old likelihoods, a bound $\|\psi_t(\mathcal{R}, \theta)\| \le M_t$ gives

$$\|g_t(P_t, \theta) - g_t(Q_t, \theta)\| \le 2M_t\,\mathcal{S}_t. \tag{4}$$

Router staleness therefore directly controls the gradient deviation of replay (Appendix B.1). Section 4 turns staleness into a trainable KL objective through layer-wise categorical distributions, and the top-k results in Appendix B.2 connect that objective to expert selection.

4 PR^2: Predictive Routing Replay

PR^2 selects a replay route by predicting the router's evolution, so the replayed route stays fixed within a batch while tracking router drift across batches. A per-layer linear predictor adds a learned bias when expert sets are recorded; training then replays those sets with the current model. Figure 2 summarizes this predict–record–replay cycle.

4.1 Predictive Replay Route

Route Recording. For PR^2, the rollout engine first generates a batch with behavior policy μ. The training engine then evaluates its old likelihoods and records predicted routes. Within this batch,

Algorithm 1 Predictive Routing Replay (PR^2)

1: Generate a rollout batch with μ.
2: Record $\nu, \hat{\mathcal{I}}$ and retained (h_0, z_0) using $W_{p,0}$.
3: **for** $i = 0, \ldots, \kappa - 1$ **do**
4: Evaluate mini-batch i with θ_i and cached $\hat{\mathcal{I}}$.
5: Compute the base policy loss from the recorded likelihoods.
6: **if** $i > 0$ and retained features are available **then**
7: Compute $\mathcal{L}_i$ with detached router targets.
8: Update $W_{p,i}$ using predictor learning-rate multiplier α.
9: **end if**
10: Update θ_i to θ_{i+1} with the base policy loss.
11: **end for**
12: Use the learned predictor when recording the next rollout batch.

index 0 denotes recording and i denotes an inner update. At token t and layer l, cache the router input $h_{0,t}^{(l)}$ and logits $z_{0,t}^{(l)} = h_{0,t}^{(l)} W_0^{(l)}$ (the $h_{\text{old}}, p_{\text{old}}$ of Figure 2). The recording-time predictor $W_{p,0}^{(l)}$ gives

$$\begin{aligned} q_{0,t}^{(l)} &= \mathrm{Softmax}\Big(z_{0,t}^{(l)} + h_{0,t}^{(l)} W_{p,0}^{(l)}\Big), \\ \hat{\mathcal{I}}_t^{(l)} &= \mathrm{TopK}(q_{0,t}^{(l)}, k), \qquad \hat{\rho}_t^{(l)} := q_{0,t}^{(l)}. \end{aligned} \tag{5}$$

These expert sets define the batch's fixed replay route $\hat{\mathcal{R}}$.

Replay and Expert Weights. Let $G_j(z, S)$ be the backbone's combination weight for expert j on selected set S, including its normalization and scaling. For example, $G_j(z, S) = \mathrm{Softmax}(z)_j$ for unnormalized softmax gating, or $\mathrm{Softmax}(z)_j / \sum_{a \in S} \mathrm{Softmax}(z)_a$ for selected-expert normalization. The routed outputs at recording and update i are

$$o_{0,t}^{(l)} = \sum_{j \in \hat{\mathcal{I}}_t^{(l)}} G_j\Big(z_{0,t}^{(l)} + h_{0,t}^{(l)} W_{p,0}^{(l)}, \hat{\mathcal{I}}_t^{(l)}\Big)\, E_{0,j}^{(l)}(h_{0,t}^{(l)}), \tag{6}$$

$$o_{i,t}^{(l)} = \sum_{j \in \hat{\mathcal{I}}_t^{(l)}} G_j(z_{i,t}^{(l)}, \hat{\mathcal{I}}_t^{(l)}) E_{i,j}^{(l)}(h_{i,t}^{(l)}). \tag{7}$$

Shared-expert contributions follow the backbone's usual computation. Both training-engine likelihood evaluations use $\hat{\mathcal{R}}$; the current evaluation uses the router and experts at θ_i. Its snapshot ratio is $r_{i,t} = \pi_{\theta_i}^{\hat{\mathcal{R}}}(y_t \mid s_t)/\nu_t$, with ν_t fixed at recording. Appendix C specifies the behavior correction and PR^3, which predicts and records routes in the rollout engine. Setting $W_p = 0$ recovers the corresponding replay baseline.

4.2 Evolution Predictor Training

Learning from Router Updates. The updated predictor is evaluated on the cached features:

$$q_{i,t}^{(l)} = \mathrm{Softmax}\Big(z_{0,t}^{(l)} + h_{0,t}^{(l)} W_{p,i}^{(l)}\Big), \qquad \rho_{i,t}^{(l)} = \mathrm{Softmax}(z_{i,t}^{(l)}). \tag{8}$$

Here $\rho_{i,t}^{(l)}$ is the current router's preference on the replay forward. Let $\mathbb{E}_i$ average over retained valid token positions in the current mini-batch. We minimize the layer-summed loss

$$\mathcal{L}_i := \mathcal{L}_{\mathrm{Pr}^2,i} = \sum_{l=1}^{L} \mathbb{E}_i D_{\mathrm{KL}}\Big(\mathrm{sg}(\rho_{i,t}^{(l)}) \,\Big\|\, q_{i,t}^{(l)}\Big). \tag{9}$$

The stop-gradient operator sg and detached cached features direct this loss to W_p; the policy follows the base RL objective. The cached indices remain fixed throughout the batch, and predictor updates affect the next recording phase.

From Fit to Recorded-Route Accuracy. The recording distribution q_0 and the fitted distribution q_i are connected by predictor drift $\mathcal{D}_i = \sum_l \mathbb{E}_i D_{\mathrm{KL}}(q_{i,t}^{(l)} \| q_{0,t}^{(l)})$. Let $P_{i,t} = \prod_l \rho_{i,t}^{(l)}$ and $Q_{0,t} = \prod_l q_{0,t}^{(l)}$ be the layer-wise categorical route laws. For a common fixed-route gradient kernel bounded by M, Proposition B.2 gives

$$\mathbb{E}_i \| g_t(P_{i,t}, \theta_i) - g_t(Q_{0,t}, \theta_i) \| \leq M \left(\sqrt{2\mathcal{L}_i} + \sqrt{2\mathcal{D}_i} \right). \tag{10}$$

Minimizing $\mathcal{L}_i$ therefore directly tightens the gradient deviation of the recorded route; the remaining drift term $\mathcal{D}_i$ is bounded by the squared predictor-logit change (Lemma B.3).

For the executed top-k sets, Lemma B.4 bounds the current-router mass missed by $\hat{\mathcal{I}}$ by $\sqrt{2D_{\mathrm{KL}}(\rho_i \| q_0)}$. Theorem B.6 identifies the *exact* KL radius within which the support stays unchanged, sharpening the margin certificate of Corollary B.5. When these local certificates hold throughout a computation, Corollary B.7 establishes equality of replay and fresh-route outputs and of their fixed-batch loss gradients.

Update Schedule. The rollout batch is split into κ update mini-batches. Predictor supervision starts after the first inner policy update, and a dedicated learning-rate multiplier α controls adaptation speed. Algorithm 1 gives the batch schedule; Appendix D specifies token masks, feature retention and parameter updates.

5 Experiments

We evaluate downstream reasoning, training stability, route tracking and expert usage across three MoE models and three rollout-reuse settings.

5.1 Implementation Details

Models and Training Data. We evaluate PR^2 in three MoE-based LLM settings. Qwen3-30B-A3B-Base (Yang et al., 2025) is trained on DAPO-17K (Yu et al., 2025), Moonlight-16B-A3B (Liu et al., 2025a) is trained on GSM8K (Cobbe et al., 2021), and OLMoE-1B-7B (Muennighoff et al., 2024) is trained on the RLVR-GSM dataset[1] following its original implementation. Unless otherwise stated, all methods are implemented in the VeRL (Sheng et al., 2025) framework. Each evolution predictor is one linear map per MoE layer and adds 0.19% to 0.69% of the active-expert forward FLOPs per token during route recording (Appendix E.2).

Baselines and Off-Policy Strength. We implement PR^2 on top of GRPO (Shao et al., 2024), and compare it with GSPO (Zheng et al., 2025b) and routing replay (Zheng et al., 2024) under the same settings. Rollout routing replay (Ma et al., 2025) is compared with PR^3 in Appendix F. We denote each off-policy setting as off-κ, where $\kappa = \frac{\mathcal{B}_{\text{global}}}{\mathcal{B}_{\text{update}}}$ is the ratio between the rollout batch size and the training update batch size. Each rollout batch is partitioned into κ update mini-batches; larger κ produces more policy updates before the next rollout.

Evaluation Benchmarks. For the main Qwen3-30B-A3B-Base setting, we evaluate off-2, off-4, and off-8 performance on AIME24 and AIME25 (Zhang & Math-AI, 2025), AMC23 (Mathematical Association of America, 2026), and HMMT25 (Balunović et al., 2025). For Moonlight-16B-A3B and OLMoE-1B-7B, we report GSM8K and MATH500 (Hendrycks et al., 2021) results in the cross-model evaluation. Appendix E provides detailed implementation settings.

5.2 Downstream Reasoning Accuracy

Main Comparison. Table 1 reports the main comparison on Qwen3-30B-A3B-Base under off-2, off-4, and off-8. We compare PR^2 with GRPO, GSPO, and routing replay. PR^2 achieves the best average accuracy in all three settings, reaching 46.80% under off-2, 46.24% under off-4, and 41.90% under off-8. It improves average accuracy over routing replay by 9.25, 13.00 and 8.16 points and over GSPO by 11.04, 9.73 and 8.74 points, respectively. PR^2 wins at every reuse factor, and its margin over routing replay peaks at 13.00 points under off-4, where routing replay falls below GSPO. Since PR^2 keeps the GRPO objective and changes only which expert sets are replayed, this gain comes from predictive route selection.

[1] https://huggingface.co/datasets/allenai/RLVR-GSM

Policy	Method	AIME24 (Avg@32)	AIME25 (Avg@32)	AMC23 (Avg@16)	HMMT25 (Avg@16)	Average
Off-2	GRPO	31.04	24.68	74.06	9.16	34.74
	GSPO	30.42	23.54	77.18	11.88	35.76
	GRPO + R^2	35.73	25.73	77.50	11.25	37.55
	GRPO + PR^2	**47.71**	**32.81**	**87.50**	**19.17**	**46.80**
Off-4	GRPO	25.42	17.71	70.93	8.33	30.60
	GSPO	32.50	22.08	79.38	12.08	36.51
	GRPO + R^2	28.13	21.25	73.59	10.00	33.24
	GRPO + PR^2	**47.40**	**31.67**	**85.47**	**20.42**	**46.24**
Off-8	GRPO	25.00	15.93	72.81	7.50	30.31
	GSPO	30.93	22.18	72.03	7.50	33.16
	GRPO + R^2	28.02	21.77	76.88	8.30	33.74
	GRPO + PR^2	**40.31**	**28.54**	**83.13**	**15.63**	**41.90**

Table 1: **Downstream reasoning accuracy on Qwen3-30B-A3B-Base.** R^2 refers to routing replay for simplicity. Bold values mark the best accuracy in each metric column within an off-policy strength, and shaded rows mark PR^2.

Policy	Method	Moonlight-16B-A3B GSM8K (Avg@1)	Moonlight-16B-A3B MATH500 (Pass@4)	Moonlight-16B-A3B Average	OLMoE-1B-7B GSM8K (Avg@1)	OLMoE-1B-7B MATH500 (Avg@4)	OLMoE-1B-7B Average
Off-2	GRPO	81.80	41.60	61.70	**73.76**	20.90	47.33
	GSPO	81.50	51.00	66.25	73.16	20.95	47.06
	GRPO + R^2	82.79	52.60	67.69	72.56	20.80	46.68
	GRPO + PR^2	**83.17**	**54.60**	**68.88**	73.47	**21.45**	**47.46**
Off-4	GRPO	62.09	36.40	49.25	72.48	20.15	46.32
	GSPO	62.85	37.80	50.33	**73.09**	22.05	47.57
	GRPO + R^2	73.69	49.20	61.45	72.71	21.85	47.28
	GRPO + PR^2	**74.07**	**51.20**	**62.64**	72.78	**23.00**	**47.89**
Off-8	GRPO	58.07	39.60	48.84	70.66	20.80	45.73
	GSPO	62.78	38.80	50.79	73.39	20.75	47.07
	GRPO + R^2	59.29	40.80	50.04	**73.92**	20.90	47.41
	GRPO + PR^2	**63.76**	**43.20**	**53.48**	72.71	**22.50**	**47.60**

Table 2: **Downstream reasoning accuracy on Moonlight-16B-A3B and OLMoE-1B-7B.** Columns are grouped by model. R^2 refers to routing replay for simplicity. Bold values mark the best accuracy in each metric column within an off-policy strength, and shaded rows mark PR^2.

Cross-Model Evaluation. Table 2 extends the comparison to Moonlight-16B-A3B and OLMoE-1B-7B under matched off-policy strengths. On Moonlight-16B-A3B, GRPO + PR^2 obtains the best average accuracy in all three settings, improving over routing replay by 1.19, 1.19 and 3.44 points under off-2, off-4 and off-8, with the largest gain under the strongest reuse. On OLMoE-1B-7B, PR^2 also achieves the highest average in all three settings, with gains of 0.78, 0.61 and 0.19 points over routing replay. PR^2 thus achieves the best average accuracy in all nine backbone–reuse comparisons, with the largest gains on Qwen3-30B-A3B-Base. Appendix G reports repeated Moonlight-16B-A3B runs in Miles, including math and instruction-following evaluations. Composition with GSPO raises both reported mean scores (Appendix G); the linear predictor gives higher MATH500 Pass@4 in both architecture-comparison pairs (Appendix H).

5.3 Training Stability

Training Dynamics. Under off-2 reuse on Qwen3-30B-A3B-Base (Figure 3), GRPO shows policy-gradient loss spikes and reward drops, and routing replay still oscillates late in training, whereas PR^2 keeps clipping rates substantially lower and yields smoother reward growth (Appendix H).

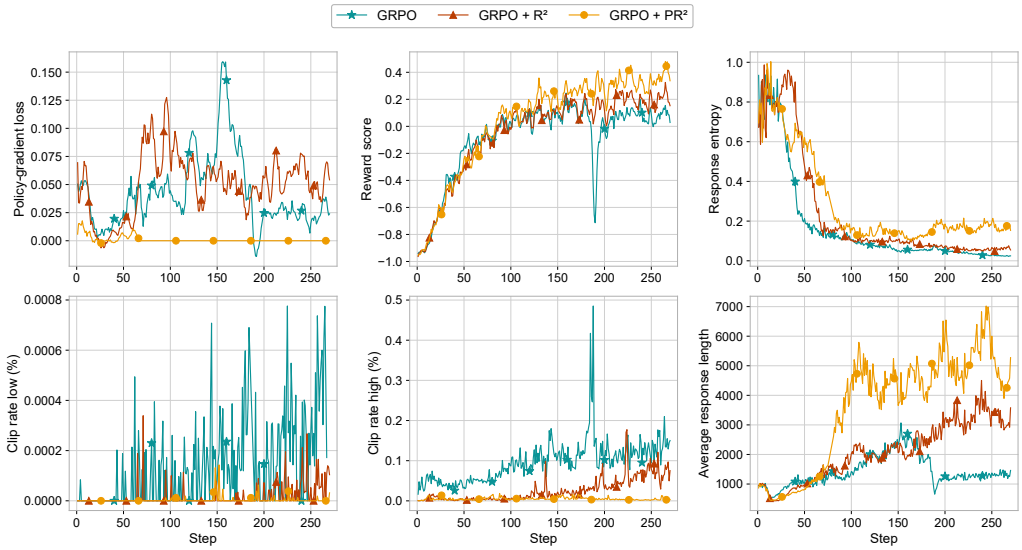
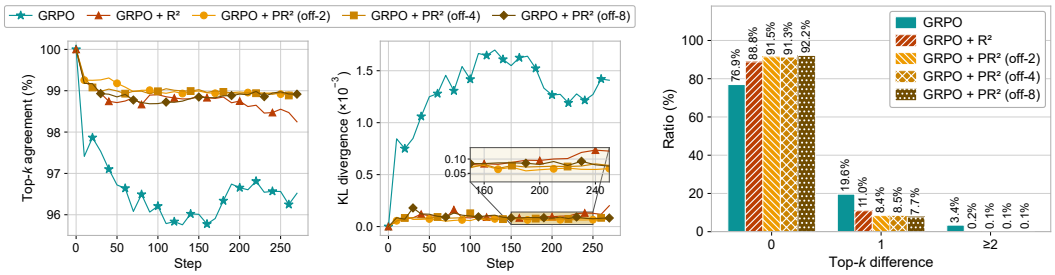


Figure 3: **Training dynamics on Qwen3-30B-A3B-Base under off-2.** We compare GRPO, routing replay, and PR^2 on policy-gradient loss, reward, response entropy, clipping rates, and response length. Smoother curves and lower clipping volatility indicate more stable updates.

5.4 Routing Prediction Analysis

Top-k Agreement and Route KL. Figure 4a reports top-k agreement and route KL on Qwen3-30B-A3B-Base. These two measurements are directly tied to the behavior of the evolution predictor. We compute agreement as $|\hat{\mathcal{I}}_t^{(l)} \cap \mathcal{I}_t^{(l)}| \,/\, k$ and average over tokens and layers. Vanilla GRPO shows a clear drop in top-k agreement and a rapid increase in route KL, indicating severe router drift during training. Routing replay reduces this drift but still becomes stale in the later stage. In contrast, PR^2 maintains higher agreement and lower KL across off-2, off-4, and off-8: the predicted replay routes track short-horizon router evolution under repeated rollout reuse.



(a) Top-k agreement and KL divergence curves.

(b) Routing deviation distribution.

Figure 4: **Routing prediction behavior during training on Qwen3-30B-A3B-Base. (a)** Top-k agreement between cached and current routing indices (left) and route KL divergence (right; the inset enlarges late training), averaged over MoE layers. Higher agreement and lower KL indicate better staleness control by PR^2. **(b)** Routing deviation distribution at the final step. More mass at 0 difference and less mass at ≥ 1 differences indicate better route tracking. Baselines are from off-2 runs unless otherwise specified.

Deviation Tails. Figure 4b measures the full distribution of route changes through the deviation count $k - |\hat{\mathcal{I}}_t^{(l)} \cap \mathcal{I}_t^{(l)}|$ on Qwen3-30B-A3B-Base. Compared with GRPO, PR^2 shifts substantially more mass to zero deviation. The zero-deviation ratio increases from 76.9% for GRPO and 88.8% for routing replay to 91.5%, 91.3%, and 92.2% under off-2, off-4, and off-8, respectively. Correspondingly, the 1-slot mismatch ratio decreases from 19.6% and 11.0% to 8.4%, 8.5%, and 7.7%, while the fraction of cases with ≥ 2 mismatched slots is reduced to only 0.1%. PR^2 thus nearly eliminates multi-slot route changes and cuts single-slot mismatches by more than half relative to GRPO.

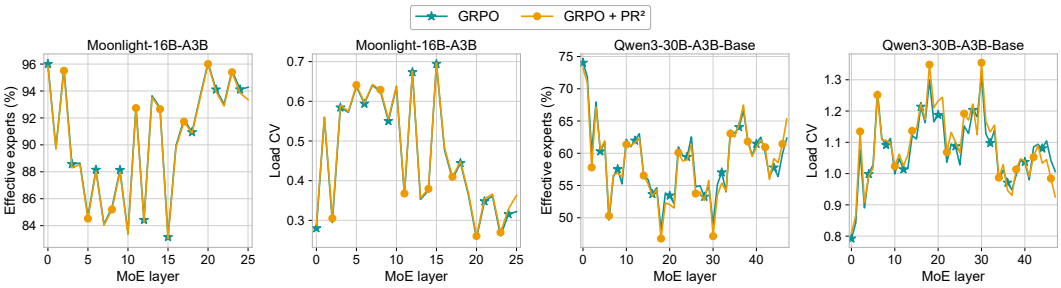


Figure 5: **Per-layer expert load of GRPO and GRPO + PR^2.** Effective experts $\exp(H)/N$ and load coefficient of variation (CV) per MoE layer on Moonlight-16B-A3B and Qwen3-30B-A3B-Base, where H is the entropy of the layer's expert-assignment shares. Lines are means over saved training steps.

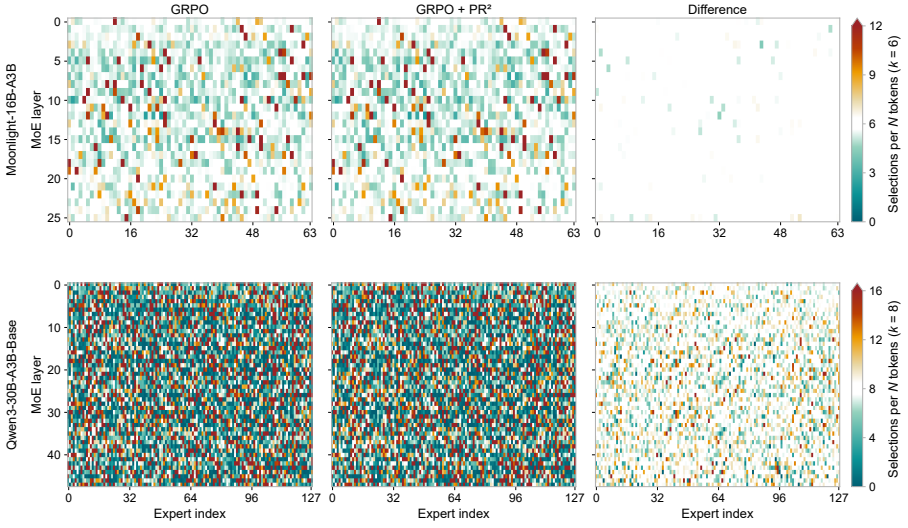


Figure 6: **Whole-model expert load, layer by expert.** Top: Moonlight-16B-A3B ($N = 64$, $k = 6$). Bottom: Qwen3-30B-A3B-Base ($N = 128$, $k = 8$). Left and centre: top-k selections of each expert per N routed tokens, pooled over saved steps, so that a uniformly used expert has value k on a linear scale from 0 to $2k$. Right: GRPO + PR^2 minus GRPO, drawn as k plus the difference on the same scale.

Expert Load. PR^2 does not worsen expert load imbalance. Across the whole model, GRPO + PR^2 keeps the layer-by-expert load pattern of GRPO (Figure 6), and its per-layer effective experts and load CV closely follow those of GRPO on Moonlight-16B-A3B and Qwen3-30B-A3B-Base (Figure 5); the step-averaged effective-expert fractions of the two methods differ by less than 0.3 percentage points. The accuracy gains of PR^2 thus come without any side effect on expert usage (Appendix I)..

6 Conclusion

This work addresses *router staleness* in MoE reinforcement learning, where replayed expert sets become outdated as the policy evolves. We introduce Predictive Routing Replay (PR^2), which uses lightweight evolution predictors to select expert sets that anticipate subsequent router updates while holding those sets fixed within each training batch. Our analysis bounds recorded-route sensitivity through predictor fit and update drift, derives an exact KL threshold for top-k agreement, and connects local support guarantees to matching forward computations and fixed-batch policy gradients. Across three MoE models and reuse factors 2, 4, and 8, PR^2 achieves the highest average accuracy in all nine comparisons, improving average math accuracy over routing replay by 8.16 to 13.00 percentage points on Qwen3-30B-A3B-Base, providing a promising alternative to vanilla routing replay.

AI USE STATEMENT

We used generative AI tools to polish the writing of this paper and run the experiments. The authors take full responsibility for all content.

ETHICS STATEMENT

This work studies reinforcement-learning optimization for publicly released MoE language models using public mathematical-reasoning and instruction-following datasets and benchmarks. Model and data use follows their published licenses. More effective post-training can support both beneficial and harmful applications of language models; downstream use should retain the base models' safety evaluations and application-specific safeguards.

REPRODUCIBILITY STATEMENT

Section 4 and Appendix D specify the generation, route-recording and training phases, including cached features, token masks and predictor updates. Appendix B gives the theorem assumptions and complete proofs. Appendix E lists model and dataset choices, optimizer settings, batch sizes, reuse factors and evaluation settings; Appendix E.2 provides the cache shapes and analytical projection cost. Main results use VeRL. The repeated-run, GSPO-composition and predictor-architecture studies use Miles, with matched comparisons and run counts specified in Appendix G.

REFERENCES

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. GPT-oss-120b and GPT-oss-20b model card. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2508.10925.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. MathArena, 2025. URL https://matharena.ai/.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint (arXiv)*, 2022. URL https://arxiv.org/abs/2211.12588.

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

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. StableMoE: Stable routing strategy for mixture of experts. *arXiv preprint (arXiv)*, 2022. URL https://arxiv.org/abs/2204.08396.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. Technical report, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash/blob/main/DeepSeek_V4.pdf.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *International Conference on Machine Learning (ICML)*, 2018. URL https://proceedings.mlr.press/v80/espeholt18a.html.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research (JMLR)*, 2022. URL https://www.jmlr.org/papers/v23/21-0998.html.

Chang Gao, Chujie Zheng, Xiong-Hui Chen, Kai Dang, Shixuan Liu, Bowen Yu, An Yang, Shuai Bai, Jingren Zhou, and Junyang Lin. Soft adaptive policy optimization. *arXiv preprint (arXiv)*, 2025a. URL https://arxiv.org/abs/2511.20347.

Jiaxuan Gao, Wei Fu, Minyang Xie, Shusheng Xu, Chuyi He, Zhiyu Mei, Banghua Zhu, and Yi Wu. Beyond ten turns: Unlocking long-horizon agentic search with large-scale asynchronous RL. *arXiv preprint (arXiv)*, 2025b. URL https://arxiv.org/abs/2508.07976.

Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2507.06261.

GLM Team. GLM-5: From vibe coding to agentic engineering. *arXiv preprint (arXiv)*, 2026. URL https://arxiv.org/abs/2602.15763.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2501.12948.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint (arXiv)*, 2021. URL https://arxiv.org/abs/2103.03874.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder task needs more experts: Dynamic routing in MoE models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://aclanthology.org/2024.acl-long.696/.

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

Jaehan Kim, Minkyoo Song, Seungwon Shin, and Sooel Son. Defending MoE LLMs against harmful fine-tuning via safety routing alignment. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2509.22745.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi K2: Open agentic intelligence. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2507.20534.

Wouter Kool, Chris J. Maddison, and Andriy Mnih. Unbiased gradient estimation with balanced assignments for mixtures of experts. In *Neural Information Processing Systems Workshop (NeurIPS Workshop)*, 2021. URL https://openreview.net/forum?id=Hvfva7l1tcj.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint (arXiv)*, 2020. URL https://arxiv.org/abs/2006.16668.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning (ICML)*, 2021. URL https://proceedings.mlr.press/v139/lewis21a.html.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 technical report. *arXiv preprint (arXiv)*, 2024a. URL https://arxiv.org/abs/2412.19437.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for LLM training. *arXiv preprint (arXiv)*, 2025a. URL https://arxiv.org/abs/2502.16982.

Liyuan Liu, Jianfeng Gao, and Weizhu Chen. Sparse backpropagation for MoE training. *arXiv preprint (arXiv)*, 2023. URL https://arxiv.org/abs/2310.00811.

Liyuan Liu, Young Jin Kim, Shuohang Wang, Chen Liang, Yelong Shen, Hao Cheng, Xiaodong Liu, Masahiro Tanaka, Xiaoxia Wu, Wenxiang Hu, Vishrav Chaudhary, Zeqi Lin, Chenruidong Zhang, Jilong Xue, Hany Awadalla, Jianfeng Gao, and Weizhu Chen. GRIN: Gradient-informed MoE. *arXiv preprint (arXiv)*, 2024b. URL https://arxiv.org/abs/2409.12136.

Zhengzhong Liu, Liping Tang, Linghao Jin, Haonan Li, Nikhil Ranjan, Desai Fan, Shaurya Rohatgi, Richard Fan, Omkar Pangarkar, Huijuan Wang, et al. K2-V2: A 360-open, reasoning-enhanced LLM. *arXiv preprint (arXiv)*, 2025b. URL https://arxiv.org/abs/2512.06201.

Wenhan Ma, Hailin Zhang, Liang Zhao, Yifan Song, Yudong Wang, Zhifang Sui, and Fuli Luo. Stabilizing MoE reinforcement learning by aligning training and inference routers. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2510.11370.

Mathematical Association of America. AMC student programs. Mathematical Association of America (MAA), 2026. URL https://maa.org/student-programs/amc/.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016. URL https://proceedings.mlr.press/v48/mniha16.html.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. OLMoE: Open mixture-of-experts language models. *arXiv preprint (arXiv)*, 2024. URL https://arxiv.org/abs/2409.02060.

OpenAI. GPT-4 technical report. *arXiv preprint (arXiv)*, 2024. URL https://arxiv.org/abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint (arXiv)*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint (arXiv)*, 2024. URL https://arxiv.org/abs/2402.03300.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *European Conference on Computer Systems (EuroSys)*, 2025. URL https://arxiv.org/abs/2409.19256.

Miles Team. Miles: Enterprise-grade reinforcement learning for large-scale model post-training. https://github.com/radixark/miles, 2026.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning. GitHub repository, 2020. URL https://github.com/huggingface/trl.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2505.09388.

Feng Yao, Liyuan Liu, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. Your efficient RL framework secretly brings you off-policy RL training. Feng Yao's Notion, 2025. URL https://fengyao.notion.site/off-policy-rl.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2503.14476.

Di Zhang, Xun Wu, Shaohan Huang, Yaru Hao, Li Dong, Zewen Chi, Zhifang Sui, and Furu Wei. Towards stable and effective reinforcement learning for mixture-of-experts. *arXiv preprint (arXiv)*, 2025. URL https://arxiv.org/abs/2510.23027.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (AIME). Mathematical Association of America (MAA), 2025. URL https://maa.org/maa-invitational-competitions/.

Yuzhong Zhao, Yue Liu, Junpeng Liu, Jingye Chen, Xun Wu, Yaru Hao, Tengchao Lv, Shaohan Huang, Lei Cui, Qixiang Ye, Fang Wan, and Furu Wei. Geometric-mean policy optimization. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=nCEs0tSwc2.

Chujie Zheng, Kai Dang, Bowen Yu, Mingze Li, Huiqiang Jiang, Junrong Lin, Yuqiong Liu, Hao Lin, Chencan Wu, Feng Hu, et al. Stabilizing reinforcement learning with LLMs: Formulation and practices. *arXiv preprint (arXiv)*, 2025a. URL https://arxiv.org/abs/2512.01374.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint (arXiv)*, 2025b. URL https://arxiv.org/abs/2507.18071.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Livia Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, et al. SGLang: Efficient execution of structured language model programs. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. LLaMA-MoE: Building mixture-of-experts from LLaMA with continual pre-training. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024. URL https://aclanthology.org/2024.emnlp-main.890/.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An LLM post-training framework for RL scaling. GitHub repository, 2025. URL https://github.com/THUDM/slime.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. ST-MoE: Designing stable and transferable sparse expert models. *arXiv preprint (arXiv)*, 2022. URL https://arxiv.org/abs/2202.08906.

A RL OBJECTIVES AND NOTATION

A.1 NOTATION AND SETUP

For a fixed rollout batch, $\mu_t = \mu(y_t \mid x, y_{<t})$ is the generating behavior likelihood, ν_t is the old likelihood recorded by the training engine, and $\pi^{\hat{\mathcal{R}}}_{\theta,t} = \pi_\theta(y_t \mid x, y_{<t}, \hat{\mathcal{R}}_{\le t})$ is the current replay likelihood. Recorded likelihoods, advantages and cached routes are held fixed during each policy update. The surrogates below are maximized; a loss-minimizing implementation uses their negatives.

A.2 PROXIMAL POLICY OPTIMIZATION (PPO)

The snapshot ratio and the token-clipped surrogate (Schulman et al., 2017) are

$$r_t(\theta) = \frac{\pi^{\hat{\mathcal{R}}}_{\theta,t}}{\nu_t}, \tag{11}$$

$$\mathcal{L}_{\mathrm{PPO}}(\theta) = \mathbb{E}_{\mathrm{batch}}\left[\sum_t \min\left\{r_t(\theta)A_t,\ \mathrm{Clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right\}\right]. \tag{12}$$

The expectation denotes the empirical batch average with the base optimizer's token masks. A reference-policy penalty can be added as

$$\mathcal{L}_{\mathrm{PPO+KL}}(\theta) = \mathcal{L}_{\mathrm{PPO}}(\theta) - \beta\mathbb{E}_{\mathrm{batch}}\left[\sum_t D_{\mathrm{KL}}\Big(\pi^{\hat{\mathcal{R}}}_\theta(\cdot \mid s_t) \,\Big\|\, \pi_{\mathrm{ref}}(\cdot \mid s_t)\Big)\right]. \tag{13}$$

Appendix C factors the behavior likelihood into the snapshot ratio and the backend's rollout correction.

A.3 GROUP RELATIVE POLICY OPTIMIZATION (GRPO)

GRPO (Shao et al., 2024; Guo et al., 2025) forms advantages from G completions of the same prompt:

$$A^{(g)}_t = \frac{R(x, y^{(g)}) - \frac{1}{G}\sum_{g'=1}^{G} R(x, y^{(g')})}{\mathrm{Std}(\{R(x, y^{(g')})\}_{g'=1}^{G}) + \epsilon_A}. \tag{14}$$

We use the Clip-Higher form of DAPO (Yu et al., 2025), with lower and upper clipping widths ϵ_{low} and ϵ_{high}:

$$\mathcal{L}^{\mathrm{DAPO}}_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{\mathrm{batch}}\left[\frac{1}{G}\sum_{g=1}^{G}\sum_t \min\Big\{r^{(g)}_t(\theta)A^{(g)}_t,\ \mathrm{Clip}\Big(r^{(g)}_t(\theta), 1-\epsilon_{\mathrm{low}}, 1+\epsilon_{\mathrm{high}}\Big)A^{(g)}_t\Big\}\right]. \tag{15}$$

The predictor is optimized by its separate loss in Eq. (9); these policy-surrogate formulas are inherited from the base optimizer.

B THEORETICAL DETAILS

B.1 ROUTING REPLAY GRADIENT DEVIATION BOUND

All divergences use natural logarithms. For probability vectors P, Q on a finite route space Ω, define $D_{\mathrm{TV}}(P, Q) = \frac{1}{2}\sum_{\omega\in\Omega}|P(\omega) - Q(\omega)|$. At fixed parameters and token context, let $\psi_t(\omega, \theta)$ be a vector-valued fixed-route gradient contribution, and set

$$g_t(P, \theta) = \sum_{\omega\in\Omega} P(\omega)\psi_t(\omega, \theta). \tag{16}$$

When ψ_t is a PPO or GRPO gradient, both evaluations use the same frozen old likelihood, advantage, correction weight and prefix.

Lemma B.1 (Sensitivity to the route distribution)**.** *For any norm and a common kernel satisfying* $\|\psi_t(\omega,\theta)\| \le M_t$ *for every* $\omega \in \Omega$*,*

$$\|g_t(P,\theta) - g_t(Q,\theta)\| \le 2M_t D_{\mathrm{TV}}(P,Q). \tag{17}$$

Proof. The triangle inequality for the finite sum gives

$$\left\| \sum_{\omega} (P(\omega) - Q(\omega))\psi_t(\omega,\theta) \right\| \le \sum_{\omega} |P(\omega) - Q(\omega)|\, M_t = 2M_t D_{\mathrm{TV}}(P,Q).$$

□

B.2 Predictor Fit, Update Drift and Recorded Routes

Fix an inner update i and a nonempty finite set $\mathcal{C}_i$ of retained valid token positions. Write $\mathbb{E}_i$ for its normalized empirical average. Every loss in this subsection uses this same token measure and sums over layers. The router probabilities $\rho_{i,t}^{(l)}$ come from the replay forward at θ_i. The distributions $q_{0,t}^{(l)}$ and $q_{i,t}^{(l)}$ use the recording-time and current predictor, respectively, on the same cached features, as in Eqs. (5) and (8).

The categorical route variable $Z_t = (Z_t^{(1)}, \dots, Z_t^{(L)}) \in \{1,\dots,N\}^L$ has product laws

$$P_{i,t}(Z_t) = \prod_l \rho_{i,t}^{(l)}(Z_t^{(l)}), \quad Q_{i,t}(Z_t) = \prod_l q_{i,t}^{(l)}(Z_t^{(l)}), \quad Q_{0,t}(Z_t) = \prod_l q_{0,t}^{(l)}(Z_t^{(l)}). \tag{18}$$

This construction samples one expert label per layer using probabilities evaluated on the fixed replay computation. Define recorded-route KL, fitting KL and predictor drift by

$$\mathcal{K}_i = \sum_l \mathbb{E}_i D_{\mathrm{KL}}(\rho_{i,t}^{(l)} \| q_{0,t}^{(l)}), \quad \mathcal{L}_i = \sum_l \mathbb{E}_i D_{\mathrm{KL}}(\rho_{i,t}^{(l)} \| q_{i,t}^{(l)}), \quad \mathcal{D}_i = \sum_l \mathbb{E}_i D_{\mathrm{KL}}(q_{i,t}^{(l)} \| q_{0,t}^{(l)}). \tag{19}$$

Proposition B.2 (Recorded-route sensitivity with predictor updates)**.** *For finite logits, the product laws in Eq.* (18)*, and a common fixed-route kernel with* $\|\psi_t(Z,\theta_i)\| \le M$ *for all* $t \in \mathcal{C}_i$ *and* $Z \in \{1,\dots,N\}^L$*,*

$$\mathbb{E}_i \|g_t(P_{i,t},\theta_i) - g_t(Q_{0,t},\theta_i)\| \le M \min\left\{ \sqrt{2\mathcal{K}_i}, \sqrt{2\mathcal{L}_i} + \sqrt{2\mathcal{D}_i} \right\}. \tag{20}$$

The same kernel gives $\mathbb{E}_i \|g_t(P_{i,t},\theta_i) - g_t(Q_{i,t},\theta_i)\| \le M\sqrt{2\mathcal{L}_i}$*.*

Proof. Finite softmax logits give strictly positive distributions. Expanding the logarithm of the product and summing the other coordinates yields

$$D_{\mathrm{KL}}(P_{i,t} \| Q_{i,t}) = \sum_l D_{\mathrm{KL}}(\rho_{i,t}^{(l)} \| q_{i,t}^{(l)}),$$

and the same identity holds for $(P_{i,t}, Q_{0,t})$ and $(Q_{i,t}, Q_{0,t})$. Lemma B.1, Pinsker's inequality and the uniform kernel bound imply

$$\|g_t(P_{i,t},\theta_i) - g_t(Q_{0,t},\theta_i)\| \le M\sqrt{2D_{\mathrm{KL}}(P_{i,t} \| Q_{0,t})}.$$

Taking the finite empirical average and using concavity of the square root gives the first branch, $M\sqrt{2\mathcal{K}_i}$. For the second branch, use the triangle inequality in total variation:

$$\begin{aligned} \|g_t(P_{i,t},\theta_i) - g_t(Q_{0,t},\theta_i)\| &\le 2M \left(D_{\mathrm{TV}}(P_{i,t}, Q_{i,t}) + D_{\mathrm{TV}}(Q_{i,t}, Q_{0,t}) \right) \\ &\le M\sqrt{2D_{\mathrm{KL}}(P_{i,t} \| Q_{i,t})} + M\sqrt{2D_{\mathrm{KL}}(Q_{i,t} \| Q_{0,t})}. \end{aligned}$$

Apply the empirical-average square-root inequality separately to the two terms and use the product identities. Both branches hold, so their minimum also bounds the left side. Applying the first argument to $(P_{i,t}, Q_{i,t})$ proves the final statement. □

Lemma B.3 (Predictor updates bound distribution drift)**.** *Let $\Delta W_p^{(l)} = W_{p,i}^{(l)} - W_{p,0}^{(l)}$. For the cached features and empirical measure above,*

$$\mathcal{D}_i \le \frac{1}{4}\sum_l \mathbb{E}_i\|h_{0,t}^{(l)}\Delta W_p^{(l)}\|_2^2 \le \frac{1}{4}\sum_l \|\Delta W_p^{(l)}\|_{\mathrm{op}}^2 \mathbb{E}_i\|h_{0,t}^{(l)}\|_2^2. \tag{21}$$

Proof. For $F(u) = \log\sum_j e^{u_j}$, write $s = \nabla F(u) = \mathrm{Softmax}(u)$. Its Hessian is $\nabla^2 F(u) = \mathrm{diag}(s) - ss^\top$. For any vector v, the quadratic form is the variance of a scalar taking values v_j with probabilities s_j. If $a = \min_j v_j$, $b = \max_j v_j$, and $m = \sum_j s_j v_j$, the inequality $(v_j - a)(b - v_j) \ge 0$ implies

$$v^\top \nabla^2 F(u) v \le (m-a)(b-m) \le \frac{(b-a)^2}{4} \le \frac{\|v\|_2^2}{2}.$$

The last step follows from $(v_j - v_r)^2 \le 2(v_j^2 + v_r^2)$ for indices attaining the extremes. Thus $\|\nabla^2 F(u)\|_{\mathrm{op}} \le 1/2$ uniformly in u and N. The integral remainder formula gives, for finite vectors u, v,

$$\begin{aligned} D_{\mathrm{KL}}(\mathrm{Softmax}(u)\|\mathrm{Softmax}(v)) &= F(v) - F(u) - \nabla F(u)^\top (v-u) \\ &= \int_0^1 (1-s)(v-u)^\top \nabla^2 F(u + s(v-u))(v-u)\, ds \\ &\le \tfrac{1}{4}\|v-u\|_2^2. \end{aligned}$$

Substitute $u = z_0 + h_0 W_{p,i}$ and $v = z_0 + h_0 W_{p,0}$ at each token and layer, then sum and average. The second inequality in Eq. (21) is the operator-norm bound for matrix multiplication. □

For a frozen teacher p and cached row vector h_0, the per-token loss also has the explicit derivative

$$\nabla_{W_p} D_{\mathrm{KL}}(p \,\|\, \mathrm{Softmax}(z_0 + h_0 W_p)) = h_0^\top (q - p). \tag{22}$$

Indeed, its W_p-dependent part is $F(z_0 + h_0 W_p) - p^\top (z_0 + h_0 W_p)$. Differentiation gives Eq. (22), and the positive-semidefinite Hessian of F makes this a convex fitting problem for fixed features and targets.

B.3 Hard Top-k Support Guarantees

Lemma B.4 (Predictive KL controls top-k support regret)**.** *Let p, q be probability vectors on N experts, $1 \le k \le N$, and let $S^\star = \mathrm{TopK}(p,k)$, $S = \mathrm{TopK}(q,k)$ under any tie rule. With $p(A) = \sum_{j\in A} p_j$,*

$$0 \le p(S^\star) - p(S) \le 2D_{\mathrm{TV}}(p,q) \le \sqrt{2D_{\mathrm{KL}}(p\|q)}. \tag{23}$$

For replay at update i, take $p = \rho_{i,t}^{(l)}$ and $q = q_{0,t}^{(l)}$.

Proof. Optimality of $S^\star$ gives nonnegative regret, and optimality of S for q gives $q(S^\star) - q(S) \le 0$. Consequently,

$$p(S^\star) - p(S) = p(S^\star) - q(S^\star) + q(S^\star) - q(S) + q(S) - p(S) \le 2D_{\mathrm{TV}}(p,q).$$

Here each set-mass difference is at most $D_{\mathrm{TV}}(p,q)$, including when the supports overlap. Pinsker's inequality gives the last step; when KL is infinite the bound holds in the extended real sense. □

Corollary B.5 (Margin certificate)**.** *For $1 \le k < N$, let $p_{(1)} \ge \cdots \ge p_{(N)}$ and $\Delta_k(p) = p_{(k)} - p_{(k+1)} > 0$. Then*

$$D_{\mathrm{KL}}(p\|q) < \tfrac{1}{2}\Delta_k(p)^2 \quad\Longrightarrow\quad \mathrm{TopK}(q,k) = \mathrm{TopK}(p,k) \tag{24}$$

for every tie rule.

Proof. Set $d = p - q$. For $a \in S^\star$ and $b \notin S^\star$, the singleton form of total variation and Pinsker's inequality give

$$q_a - q_b = (p_a - p_b) - (d_a - d_b) \ge \Delta_k(p) - 2D_{\mathrm{TV}}(p,q) \ge \Delta_k(p) - \sqrt{2D_{\mathrm{KL}}(p\|q)} > 0.$$

Every selected expert therefore has strictly larger q-probability than every unselected expert. □

Theorem B.6 (Exact KL radius for top-k agreement)**.** *Let p lie in the interior of the N-expert probability simplex, $1 \le k < N$, and set $a = p_{(k)} > b = p_{(k+1)} > 0$. Define*

$$\tau_k(p) = a \log \frac{2a}{a+b} + b \log \frac{2b}{a+b}. \tag{25}$$

For every fixed tie-breaking rule,

$$\inf_{\substack{q \text{ in the simplex interior} \\ \mathrm{TopK}(q,k) \neq \mathrm{TopK}(p,k)}} D_{\mathrm{KL}}(p\|q) = \tau_k(p). \tag{26}$$

Thus $D_{\mathrm{KL}}(p\|q) < \tau_k(p)$ certifies exact support agreement. In addition,

$$\tau_k(p) \ge \frac{(a-b)^2}{2(a+b)} \ge \frac{(a-b)^2}{2}, \tag{27}$$

so the exact radius strengthens the margin certificate.

Proof. Let $S^\star = \mathrm{TopK}(p,k)$. If $S = \mathrm{TopK}(q,k)$ differs from $S^\star$, some $r \in S^\star \setminus S$ and $s \in S \setminus S^\star$ satisfy $q_r \le q_s$. Write $u = p_r > v = p_s$, $\sigma = u + v$, $\eta = q_r + q_s$, $\alpha = u/\sigma > 1/2$ and $\beta = q_r/\eta \le 1/2$.

For $N > 2$, grouping the pair $\{r, s\}$ and its complement gives the exact identity

$$D_{\mathrm{KL}}(p\|q) = d_{\mathrm{B}}(\sigma\|\eta) + \sigma\, d_{\mathrm{B}}(\alpha\|\beta) + (1-\sigma) D_{\mathrm{KL}}(\bar{p}\|\bar{q}),$$

where d_{B} is Bernoulli KL and $\bar{p}, \bar{q}$ are the normalized distributions on the complement. Interior probabilities ensure all denominators are positive. Since $\partial_\beta d_{\mathrm{B}}(\alpha\|\beta) = (\beta - \alpha)/(\beta(1-\beta)) < 0$ for $0 < \beta \le 1/2$, nonnegativity of the other two KL terms yields

$$D_{\mathrm{KL}}(p\|q) \ge \sigma d_{\mathrm{B}}(\alpha\|1/2) =: c(u,v) = u \log \frac{2u}{u+v} + v \log \frac{2v}{u+v}.$$

For $N = 2$, $\sigma = \eta = 1$ and the same bound follows directly from the two-coordinate Bernoulli KL. The bound is attained on the half-space $q_r \le q_s$ at $q_r = q_s = (u+v)/2$ with all other entries unchanged.

For $u > v > 0$,

$$\partial_u c(u,v) = \log \frac{2u}{u+v} > 0, \qquad \partial_v c(u,v) = \log \frac{2v}{u+v} < 0.$$

Every cross-boundary pair has $u \ge a$ and $v \le b$, hence $c(u,v) \ge c(a,b) = \tau_k(p)$. This proves the lower bound in Eq. (26).

For the reverse bound, choose indices attaining a and b, leave all other entries equal to p, and set $q_r = (a+b)/2 - \varepsilon$, $q_s = (a+b)/2 + \varepsilon$. For sufficiently small $\varepsilon > 0$, all probabilities remain positive and $q_s > q_r$ changes the support under every tie rule. By continuity of KL, its value tends to $\tau_k(p)$ as $\varepsilon \downarrow 0$. This proves the infimum statement, including tie rules that retain $S^\star$ at the boundary itself.

Finally, Bernoulli Pinsker gives

$$\tau_k(p) = (a+b) d_{\mathrm{B}}\left(\frac{a}{a+b} \,\middle\|\, \frac{1}{2}\right) \ge 2(a+b)\left(\frac{a}{a+b} - \frac{1}{2}\right)^2 = \frac{(a-b)^2}{2(a+b)}.$$

Since $a + b \le 1$, Eq. (27) follows. □

B.4 From Local Certificates to the Complete Computation

Corollary B.7 (Certified replay equals fresh routing)**.** *Fix a finite token batch and a deterministic MoE computation graph at parameters θ. Replay and fresh routing use the same operators, gating normalization, experts and random state. Include every routing event required for the outputs, including prefix and key–value construction. Suppose each event v uses direct top-k_v selection, $1 \le k_v < N_v$, and has:*

(i) recorded distribution $q_{0,v}$ with replay set $S_v = \mathrm{TopK}(q_{0,v}, k_v)$;

(ii) current distribution p_v, evaluated on the replay input, with finite logits and positive boundary margin;

(iii) $D_{\mathrm{KL}}(p_v \| q_{0,v}) < \tau_{k_v}(p_v)$.

Then replay and fresh routing have identical outputs at every node. If the fixed-support computations and routing scores are continuous near θ, they agree on a neighborhood of θ. Consequently their fixed-batch policy-surrogate gradients agree wherever the surrogate is differentiable, with the same frozen old likelihoods, advantages and correction weights.

Proof. Order the finite graph topologically. Initial token inputs are identical. Suppose all predecessors of a node have equal outputs. At an ordinary node, identical operators give equal outputs. At a routing node v, equal predecessors give the same router input and distribution p_v in both computations. Theorem B.6 applies by (i)–(iii), so fresh top-k_v selection returns S_v. Using the same expert functions and combination weights on this support gives the same output. This proves the induction step and therefore equality throughout the graph, including the prefix states needed by later tokens.

For neighborhood equality, evaluate the graph with its recorded supports fixed. Continuity and each strict boundary margin imply an open neighborhood of θ on which that event's top-k_v set stays S_v. There are finitely many events, so the intersection of these neighborhoods is open and contains θ. At every point in the intersection, the same topological induction equates the fresh and fixed-support graphs. Applying the same fixed-batch surrogate to identical output functions gives equal loss functions on this neighborhood, and hence equal derivatives at their differentiability points. □

B.5 Numerical Check of the Top-k Bridge

The saved-logit analysis of Qwen3-30B-A3B-Base under off-4 compares current-router probabilities with the recorded predictive distributions at matched tokens across all 48 MoE layers. Lemma B.4's regret bound holds on every logged observation, and exact support agreement is 90.43%. The squared-margin certificate in Eq. (24) certifies 1.25% of observations. Equation (25) gives the strengthened analytic certificate, whose acceptance region contains that of the squared-margin rule.

C Rollout-Engine Behavior Policy

C.1 Three Passes and Likelihood Ratios

Each iteration has a generating pass, an old-logprob pass and a gradient pass. Denote the actual generating likelihood by μ_t, the recorded old likelihood by ν_t, and the current replay likelihood by $\pi^{\hat{\mathcal{R}}}_{\theta_i,t}$. Positive token probabilities give the exact factorization

$$w_{i,t} := \frac{\pi^{\hat{\mathcal{R}}}_{\theta_i,t}}{\mu_t} = \underbrace{\frac{\nu_t}{\mu_t}}_{c_t:\ \text{behavior correction}} \underbrace{\frac{\pi^{\hat{\mathcal{R}}}_{\theta_i,t}}{\nu_t}}_{r_{i,t}:\ \text{snapshot ratio}} . \tag{28}$$

This separates the behavior-to-recording change from the subsequent policy update (Zheng et al., 2025a). The base optimizer supplies its clipping and correction operator. For example, truncated importance sampling uses a capped c_t as a sample weight on a snapshot-ratio surrogate. PR^2 retains the configured operator while choosing the expert sets used to evaluate ν_t and $\pi^{\hat{\mathcal{R}}}_{\theta_i,t}$.

In PR^2, generation uses the rollout router. The old-logprob pass applies the predictor and records $\hat{\mathcal{R}}$ and ν_t using Eq. (6). The gradient pass replays $\hat{\mathcal{R}}$ with the current router weights in Eq. (7). Thus the two training-engine passes share expert indices; their likelihoods incorporate their respective gate weights and model parameters.

C.2 Rollout Routing Replay

Rollout routing replay (Ma et al., 2025) records the generating engine's expert sets and replays them in both training-engine passes. Its old-logprob and gradient evaluations use the snapshot and current

gate weights, respectively. The same factorization in Eq. (28) tracks their relation to the generating policy.

C.3 PR^3: PREDICTIVE ROLLOUT ROUTING REPLAY

PR^3 places the predictor in the rollout engine. Equations (5) and (6) then use rollout-engine features and experts, and the predicted supports drive autoregressive generation. The recorded behavior likelihood μ_t includes that predictive forward. Both training-engine passes replay the generated supports with their own gate weights and bypass the predictor. Its KL supervision uses the cached rollout features and detached current-router targets, following Eq. (9). This gives the same predict–record–replay construction for rollout routing replay; Table 4 reports the comparison.

D PREDICTIVE ROUTING REPLAY PSEUDOCODE

The batch schedule in Algorithm 1 uses mini-batch indices $i = 0, \ldots, \kappa - 1$. The predictor and model at the start of update i are $W_{p,i}$ and θ_i. Index caches cover the full trajectory; feature caches contain the retained positions described in Appendix E.2. Every auxiliary KL is normalized over the retained valid positions in its mini-batch and summed over layers.

Algorithm 2 PR^2 route recording in the training-engine old-logprob pass

Require: Token position t, layer l, input h_0, snapshot router $W_0^{(l)}$, recording predictor $W_{p,0}^{(l)}$, snapshot experts, index cache $\mathcal{P}$, feature cache $\mathcal{F}$.

$z_0 \leftarrow h_0 W_0^{(l)}$; $u_0 \leftarrow z_0 + h_0 W_{p,0}^{(l)}$.
$q_0 \leftarrow \mathrm{Softmax}(u_0)$; $S \leftarrow \mathrm{TopK}(q_0, k)$.
$\mathcal{P}[(t,l)] \leftarrow S$.
if t is retained by the feature-cache mask **then**
    $\mathcal{F}[(t,l)] \leftarrow (\mathrm{sg}(h_0), \mathrm{sg}(z_0))$.
end if
$o_0 \leftarrow \sum_{j \in S} G_j(u_0, S) E_{0,j}^{(l)}(h_0)$.
Continue the forward pass and store its token likelihood ν_t.

Algorithm 3 PR^2 replay and predictor supervision at inner update i

Require: Update i, token t, layer l, current input h_i, router $W_i^{(l)}$, predictor $W_{p,i}^{(l)}$, current experts, caches $\mathcal{P}, \mathcal{F}$.

$z_i \leftarrow h_i W_i^{(l)}$; $\rho_i \leftarrow \mathrm{Softmax}(z_i)$; $S \leftarrow \mathcal{P}[(t,l)]$.
$o_i \leftarrow \sum_{j \in S} G_j(z_i, S) E_{i,j}^{(l)}(h_i)$.
Initialize this position's auxiliary loss to zero.
if $i > 0$, t is valid, and $(t,l) \in \mathcal{F}$ **then**
    $(h_0, z_0) \leftarrow \mathcal{F}[(t,l)]$.
    $q_i \leftarrow \mathrm{Softmax}(z_0 + h_0 W_{p,i}^{(l)})$.
    $\ell_{i,t}^{(l)} \leftarrow D_{\mathrm{KL}}(\mathrm{sg}(\rho_i) \| q_i)$.
end if
When supervised, aggregate $\mathcal{L}_i = \sum_l \mathbb{E}_i \ell_{i,t}^{(l)}$ over retained valid positions; otherwise use zero auxiliary loss.
Compute the base policy loss with frozen ν, advantages and cached supports.
From these losses, update θ_i to θ_{i+1} and, when supervised, $W_{p,i}$ to $W_{p,i+1}$.
Carry W_p unchanged when supervision is skipped; retain $\mathcal{P}$ throughout the batch.

E Implementation Details

E.1 Model and Hyperparameters

All three models share a common off-policy schedule: per model we fix a rollout batch size $\mathcal{B}_{\text{global}}$ and let the per-update batch size $\mathcal{B}_{\text{update}}$ take three values to realize off-2, off-4, and off-8. Unless noted otherwise, the asymmetric clip ratios are $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$, the SGLang oversampling ratio is 0.1, and predictors are zero-initialized. Following the trade-off between the training update mini-steps and learning rate described by Ma et al. (2025), we use a smaller per-update learning rate under stronger off-policy reuse. Each rollout batch supports κ inner mini-step updates under off-κ, so a smaller per-update learning rate keeps the cumulative parameter drift over a rollout comparable across off-policy strengths.

Qwen3-30B-A3B-Base. Rollout uses $\mathcal{B}_{\text{global}} = 64$ with 8 responses per prompt and a maximum generation length of 16K tokens. We sweep $\mathcal{B}_{\text{update}} \in \{32, 16, 8\}$ for off-2, off-4, and off-8 with learning rates $\{2,\ 1.5,\ 1\} \times 10^{-6}$ and predictor learning-rate multipliers $\{10^4,\ 10^3,\ 10^3\}$. Each run uses 32 NVIDIA H200 GPUs for about 72 hours.

Moonlight-16B-A3B. Rollout uses $\mathcal{B}_{\text{global}} = 32$ with 16 responses per prompt and a maximum generation length of 1K tokens. We sweep $\mathcal{B}_{\text{update}} \in \{16, 8, 4\}$ for off-2, off-4, and off-8 with learning rates $\{5,\ 3.7,\ 2.5\} \times 10^{-7}$ and predictor learning-rate multipliers $\{10^2,\ 5 \times 10^1,\ 5 \times 10^1\}$. Each run uses 32 NVIDIA H200 GPUs for about 24 hours.

OLMoE-1B-7B. Rollout uses $\mathcal{B}_{\text{global}} = 32$ with 8 responses per prompt and a maximum generation length of 1K tokens. We sweep $\mathcal{B}_{\text{update}} \in \{16, 8, 4\}$ for off-2, off-4, and off-8 with learning rates $\{5,\ 3.7,\ 2.5\} \times 10^{-7}$ and a fixed predictor learning-rate multiplier 10^3. The clip ratios are $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.1$. Each run uses 8 NVIDIA RTX PRO 6000 GPUs for about 4 hours.

Baseline Settings. GRPO follows DAPO Clip-Higher without KL regularization. GSPO uses $\epsilon_{\text{low}} = 3 \times 10^{-4}$, $\epsilon_{\text{high}} = 4 \times 10^{-4}$, and KL coefficient 10^{-3}. Routing replay records routes from the old policy snapshot. Rollout routing replay shares PR^2's data, optimizer, off-policy schedule, and validations but replays routes produced by the rollout-engine behavior policy.

PR^2 Implementation. PR^2 adds an evolution predictor to each router without modifying the PPO objective. The added computation is a single router-side projection during route recording and the predictive-loss evaluation on a bounded feature cache during training. The full predicted top-k indices are always cached for replay. Route-recording features $(h_{\text{old}}, p_{\text{old}})$ used to train the predictor are cached only for a fixed number of tokens, with cache length $T_c = 2\text{K}$ for Qwen3-30B-A3B-Base (whose maximum generation length is $T = 16\text{K}$) and $T_c = T = 1\text{K}$ for Moonlight-16B-A3B and OLMoE-1B-7B. For Qwen3-30B-A3B-Base, uniform sub-sampling distributes the T_c retained positions across the full 16K context.

E.2 Resource Details

Model	$(L, d, N, k, s, m, T, T_c)$	Index Cache	Feature Cache	FLOPs	FFN Ratio
Qwen3	$(48, 2048, 128, 8, 0, 768, 16\text{K}, 2\text{K})$	25.2 MB	453 MB	25.2M	0.69%
Moonlight	$(26, 2048, 64, 6, 2, 1408, 1\text{K}, 1\text{K})$	0.64 MB	116 MB	6.82M	0.19%
OLMoE	$(16, 2048, 64, 8, 0, 1024, 1\text{K}, 1\text{K})$	0.52 MB	71.3 MB	4.19M	0.26%

Table 3: PR^2 cache and route-recording forward-compute overhead. Index cache uses $4TLk$ bytes for int32 expert indices, feature cache uses $T_cL(2d + 4N)$ bytes for BF16 hidden features and FP32 router logits, and forward compute reports the extra $2LdN$ FLOPs per token from the predictor projection. Cache values are per response, and FLOPs values are per token. FFN Ratio divides these FLOPs by the active expert forward cost $6L(k + s)dm$.

Predictor Parameters and Compute. Each evolution predictor is a linear map $W_p^{(l)} \in \mathbb{R}^{d\times N}$. The total predictor size is therefore LdN parameters. During route recording, PR^2 adds one predictor projection per MoE layer, or $2LdN$ extra FLOPs per generated token. Table 3 reports the resulting model-specific values. Under the gated-MLP expert form with gate, up, and down projections, the Feed-Forward Network (FFN) ratio compares this route-recording forward overhead with the active expert computation, approximated as $6L(k+s)dm$ FLOPs per token. For Moonlight-16B-A3B, $s=2$ counts shared experts, while the cached routed indices still use $k=6$. The predictor therefore adds only 0.19% to 0.69% of the active expert forward cost across the three models.

Cache Cost. We store replay expert indices as int32, so the index cache adds $4Lk$ bytes per generated token across all MoE layers and scales with the rollout length T. For predictor learning, PR^2 also caches route-recording features $(h_{\text{old}}, p_{\text{old}})$ for a fixed feature-cache length T_c, with BF16 hidden features and FP32 router logits, contributing $T_cL(2d+4N)$ bytes per response. We use $T_c = 2\text{K}$ for Qwen3-30B-A3B-Base, whose maximum generation length is 16K. For Moonlight-16B-A3B and OLMoE-1B-7B the entire generation is already 1K tokens, so no downsampling is applied and $T_c = T = 1\text{K}$. This separation keeps full expert indices cheap while bounding the feature cache under long contexts.

F Additional Results with PR^3

PR^3 Settings. Table 4 compares PR^3 with rollout routing replay on Qwen3-30B-A3B-Base. Both methods use the same data and hyperparameters. The PR^3 predictor uses learning-rate multipliers $\{10^2, 10^1, 10^1\}$ for off-2, off-4, and off-8, respectively.

Policy	Method	AIME24 (Avg@32)	AIME25 (Avg@32)	AMC23 (Avg@16)	HMMT25 (Avg@16)	Average
Off-2	GRPO + R^3	42.36	28.05	84.38	13.75	42.14
	GRPO + PR^3	**44.48**	**29.69**	**87.97**	**16.67**	**44.70**
Off-4	GRPO + R^3	41.22	28.92	**85.21**	15.56	42.73
	GRPO + PR^3	**47.19**	**32.81**	83.28	**20.83**	**46.03**
Off-8	GRPO + R^3	36.18	24.20	79.22	**11.39**	37.75
	GRPO + PR^3	**40.42**	**28.33**	**81.25**	10.00	**40.00**

Table 4: **Additional PR^3 results on Qwen3-30B-A3B-Base.** R^3 denotes rollout routing replay for simplicity. Bold values mark the better accuracy in each metric column within an off-policy strength, and shaded rows mark PR^3. Results are averaged over 3 different seeds.

PR^3 Consistently Improves Rollout Routing Replay. Across all three reuse factors, PR^3 improves the average competition-math accuracy over R^3 by 2.56, 3.30 and 2.25 points under off-2, off-4 and off-8. It leads on AIME24 and AIME25 in every setting and on 10 of the 12 benchmark–setting pairs overall, so the predicted route improves rollout-engine replay as consistently as it improves old-snapshot replay.

G Repeated Runs in the Miles Codebase

Matched Repeated Runs. Every cell of Tables 1 and 2 is a single VeRL run. To measure run-to-run variation and to test PR^2 beyond mathematical rewards, we repeat matched runs of GRPO + R^2 and GRPO + PR^2 on Moonlight-16B-A3B under off-4 in a second implementation, the Miles codebase, where the two methods of a matched pair share seed, data and schedule. The math study trains on GSM8K with three runs per method and evaluates on GSM8K and MATH500. The instruction-following study trains on the AllenAI RLVR-IFEval prompts[2] with two runs per method, rewarding a

[2] https://huggingface.co/datasets/allenai/RLVR-IFEval

response by the fraction of the prompt's verifiable constraints it satisfies, and evaluates instruction-loose compliance on the held-out IFEval and IFBench suites. It uses the Moonlight-16B-A3B off-4 hyperparameters of Appendix E with at most 2K generated tokens.

Gains Hold Across Repeated Runs. In the matched Miles runs of Table 5, GRPO + PR^2 has higher mean scores on all four benchmarks, with gains of 1.04 points on GSM8K, 1.60 on MATH500, 0.84 on IFEval and 0.29 on IFBench. On math it also lowers the run-to-run standard deviation, from 1.18 to 0.31 on GSM8K and from 2.03 to 0.92 on MATH500. The advantage of PR^2 over routing replay therefore carries over to repeated runs, a second codebase and a non-mathematical reward.

Composition with GSPO. PR^2 changes which experts are replayed and leaves the policy objective untouched, so it composes directly with sequence-level objectives such as GSPO (Zheng et al., 2025b). On Moonlight-16B-A3B in the Miles codebase, with two matched runs per method, GSPO + PR^2 improves over GSPO on both benchmarks (Table 6), by 0.23 points on GSM8K and 2.20 on MATH500 Pass@4, so the routing-side gain of PR^2 adds to that of a sequence-level objective.

Method	Math		Instruction Following	
	GSM8K (Avg@1)	MATH500 (Pass@4)	IFEval (loose)	IFBench (loose)
GRPO + R^2	$86.73_{\pm 1.18}$	$70.00_{\pm 2.03}$	$53.78_{\pm 0.08}$	$33.87_{\pm 3.08}$
GRPO + PR^2	$\mathbf{87.77}_{\pm 0.31}$	$\mathbf{71.60}_{\pm 0.92}$	$\mathbf{54.62}_{\pm 1.44}$	$\mathbf{34.16}_{\pm 5.14}$

Table 5: **Repeated runs on Moonlight-16B-A3B under off-4 in the Miles codebase.** Mean over matched runs, three per method for math and two for instruction following, with the standard deviation as subscript. R^2 denotes routing replay. Bold marks the better mean, and the shaded row marks PR^2.

Method	GSM8K (Avg@1)	MATH500 (Pass@4)
GSPO	$88.40_{\pm 0.43}$	$70.00_{\pm 3.11}$
GSPO + PR^2	$\mathbf{88.63}_{\pm 1.18}$	$\mathbf{72.20}_{\pm 1.41}$

Table 6: **Composing PR^2 with GSPO on Moonlight-16B-A3B in the Miles codebase.** Mean over two matched runs per method, with the sample standard deviation as subscript. Bold marks the better mean, and the shaded row marks PR^2.

H Additional Experimental Analysis

We report additional ablations of the predictive component and diagnostics of route prediction across update horizons. Unless noted otherwise, they are conducted on Qwen3-30B-A3B-Base.

Training Dynamics. Figure 3 compares off-2 training dynamics on Qwen3-30B-A3B-Base for GRPO, routing replay, and PR^2, summarized in the Training Stability part of Section 5. We use these curves to examine whether repeated rollout reuse only affects final accuracy or also changes the optimization trajectory. Vanilla GRPO exhibits policy-gradient loss spikes and abrupt reward drops under rollout reuse. Routing replay reduces the most severe spikes, but still shows late-stage oscillations in clipping rates and entropy. In contrast, PR^2 keeps clipping rates substantially lower and yields smoother reward growth. The response-entropy and length curves show the same pattern: PR^2 avoids abrupt entropy collapse while response length grows steadily, so the policy improves its reasoning without entering unstable high-clipping updates. Predictive routing replay thus stabilizes the optimization process.

Optimization Behavior. The curves in Figure 3 illustrate how routing mismatch affects PPO-style optimization. In off-policy MoE RL, stale routing can produce extreme token-level importance ratios, which appear as elevated clipping rates and can further lead to loss spikes and reward regressions. Routing replay mitigates the most severe instability by enforcing replay consistency, but its cached route can still become stale as repeated updates move the router. PR^2 improves the training trajectory itself by replaying predicted routes that better match short-horizon router evolution.

Predictive Objective Ablation. We compare the default loss $\mathcal{L}_i$ with a delta-matching objective on the same retained tokens:

$$\mathcal{L}^{\Delta}_{\mathrm{Pr}^2,i} = \sum_l \mathbb{E}_i D_{\mathrm{KL}}\Big(\mathrm{Softmax}(\mathrm{sg}(\Delta z^{(l)}_{i,t})) \,\Big\|\, \mathrm{Softmax}(b^{(l)}_{i,t})\Big)\,, \tag{29}$$

where $\Delta z_{i,t}^{(l)} = z_{i,t}^{(l)} - z_{0,t}^{(l)}$ and $b_{i,t}^{(l)} = h_{0,t}^{(l)} W_{p,i}^{(l)}$. The left pair of panels in Figure 7 shows similar mean-advantage and AIME24 curves for both objectives, so PR^2 is insensitive to the loss form. The standard KL objective reaches slightly higher final accuracy and is our default.

Learning-Rate Multiplier Ablation. The right pair of panels in Figure 7 compares $\alpha = 10^4$ and $\alpha = 10^6$ under off-2. Both settings optimize stably and improve AIME24 accuracy, so PR^2 is insensitive to α over two orders of magnitude. The larger multiplier learns faster early in training, whereas the default $\alpha = 10^4$ gives a smoother mean-advantage trajectory and comparable final accuracy.

Predictor Architecture. We compare the linear evolution predictor of Section 4 with a two-layer MLP predictor that maps the same router input through one hidden layer before producing the logit bias. All other settings are shared. Table 7 reports two such matched runs on Moonlight-16B-A3B in the Miles codebase. The linear predictor reaches higher MATH500 Pass@4 in both runs, by 0.80 points in run A (70.00 versus 69.20) and by 1.20 points in run B (72.00 versus 70.80). On GSM8K, the MLP gains 0.76 points in run A and ties in run B, so one linear layer suffices and extra depth brings no significant gain.

Run	Predictor	GSM8K (Avg@1)	MATH500 (Pass@4)
A	Linear	88.02	**70.00**
	MLP	**88.78**	69.20
B	Linear	88.17	**72.00**
	MLP	88.17	70.80

Table 7: **Linear versus two-layer MLP evolution predictor on Moonlight-16B-A3B in the Miles codebase.** Both arms of a run share the PR^2 training and evaluation setup. Bold marks the better value within a run, and shaded rows mark the linear predictor used in the paper.

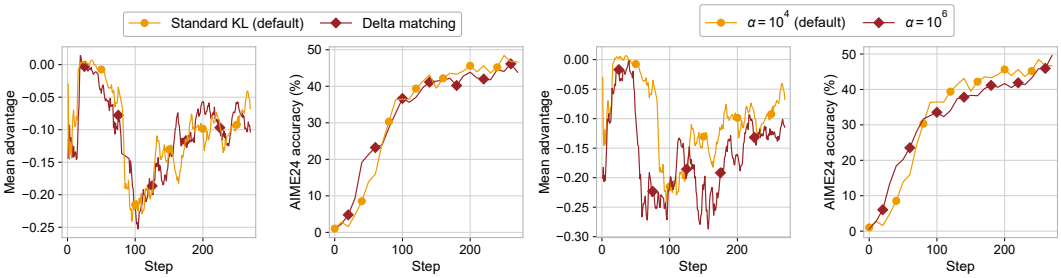


Figure 7: **Additional predictive-routing ablations on Qwen3-30B-A3B-Base under off-2.** Left pair: predictive loss form, the standard KL objective against delta matching. Right pair: predictor learning-rate multiplier α. Each pair reports the batch-averaged clipped surrogate integrand and AIME24 accuracy over training steps; the default setting is drawn in the PR^2 colour.

Layer-Wise Top-k Accuracy. Figure 8 reports prediction agreement at each MoE layer and supervised mini-step under off-2, off-4 and off-8. Most layers exceed 98% top-k agreement, and the mini-step curves largely overlap. Agreement varies more with depth, with slightly lower values in later layers. The predictor therefore remains accurate across the full within-batch update horizon.

A single linear predictor with the standard KL loss thus suffices: a deeper predictor or delta matching brings no significant gain, and agreement stays high across layers and mini-steps.

I Expert Load and Routing Specialization

We characterize expert usage and token-category routing using saved caches from independently trained GRPO and GRPO + PR^2 runs under off-2 on Moonlight-16B-A3B and Qwen3-30B-A3B-Base.

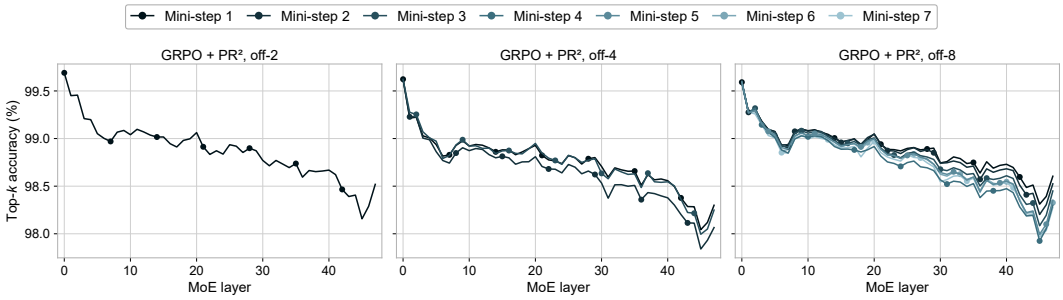


Figure 8: **Layer-wise top-k agreement across mini-steps on Qwen3-30B-A3B-Base.** The horizontal axis indexes MoE layers. Each curve reports the top-k agreement of the predicted routing at one mini-step within the batch, under off-2, off-4 and off-8; colours run from dark navy (mini-step 1) to light blue-grey (mini-step 7).

I.1 Expert Load Distribution

We compute expert load from hard top-k assignments at every saved training step. For layer assignment-share entropy H, normalized load entropy is $H/\log N$, the effective-expert fraction is $\exp(H)/N$, and load CV is the coefficient of variation of expert counts. Figures 9 and 5 show these quantities over training and by layer. On both models, the paired profiles are closely aligned: step-averaged effective-expert fractions differ by less than 0.3 percentage points. PR^2 therefore leaves expert load as balanced as under GRPO.

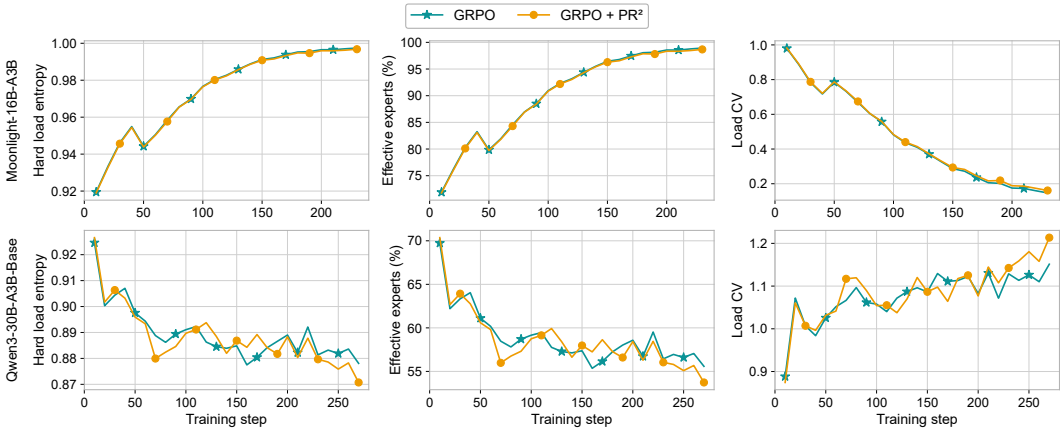


Figure 9: **Expert load over training.** Top: Moonlight-16B-A3B. Bottom: Qwen3-30B-A3B-Base. Columns: normalized hard load entropy $H/\log N$, effective experts and load CV. Lines are means over MoE layers at each saved step.

I.2 Aligned-Position Routing Specialization

Aligned Token Categories and Agreement Metrics. On Moonlight-16B-A3B, we compare one GRPO and one GRPO + PR^2 run at matched GSM8K token positions, aligned by saved token identifiers and pooled over saved steps. We group positions into six categories: prompt number, prompt text, arithmetic expression, reasoning number, reasoning text and final-answer span. For each category and layer, top-k overlap measures the fraction of shared executed experts, and exact support measures the fraction of identical expert sets. Affinity rank correlation is the Spearman correlation of per-expert selectivity $\log(P(e \mid c)/P(e \mid s))$, where s is the prompt or response segment containing category c. This normalization compares category preference relative to each expert's segment frequency.

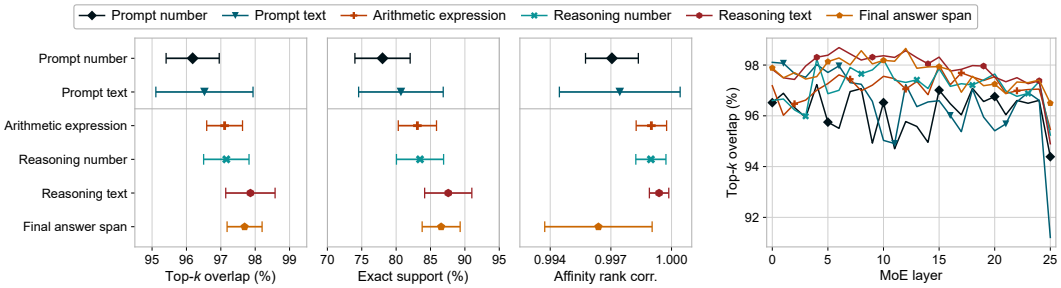


Figure 10: **Routing agreement by token category between GRPO and GRPO + PR^2 on Moonlight-16B-A3B at aligned GSM8K positions.** Left three panels: top-k overlap, exact support and affinity rank correlation per category; dots are means over the 26 MoE layers and whiskers one standard deviation across layers. Right: top-k overlap per layer, one line per token category.

Agreement Varies by Category While Load Stays Balanced. Different token categories route with measurably different agreement between the two runs (Figure 10). Prompt categories agree least, with layer-averaged top-k overlap of 96.2% for prompt numbers and 96.5% for prompt text, while reasoning text agrees most at 97.9%; exact support spans a wider range, from 78.0% for prompt numbers to 87.6% for reasoning text. Every category still keeps top-k overlap above 96% and affinity rank correlation of at least 0.996, and per-layer overlap remains high across depth, with the largest change at the final MoE layer. These category-level differences do not unbalance expert load: the load profiles of the two runs match, with step-averaged effective-expert fractions within 0.3 percentage points (Appendix I.1). PR^2 therefore preserves both the category-conditioned routing of GRPO and its balanced expert load.