Beyond the Stability-Exploration Dilemma: Environmental Regularization for LLM Policy Optimization

Xianlei Zhou[1], Xiangdi Meng[1], Yu He[1,2*], Tianyu Qi[3], Shuyan Guan[1], Xianli Zhang[1], Jian Zhang[4], Xin Li[1], Qika Lin[5], Jun Liu[2]

[1]AMAP, Alibaba Group [2]Xi'an Jiaotong University [3]JD.com
[4]Beijing Normal University [5]National University of Singapore

{zhouxianlei.zxl,gaolin.hy}@alibaba-inc.com

Abstract

Policy optimization (PO) for Large Language Models faces a stability–exploration trade-off, currently mediated by an action-side Policy-KL regularizer. This puts practitioners in a double bind: keeping Policy-KL constrains response behavior and consumes the action-side exploration budget, while dropping it leaves the optimization without an explicit drift control. We argue for an alternative that breaks the dilemma by moving regularization to the input side. As training progresses, the distribution over training queries induced by the current policy drifts unchecked from its pre-RL reference distribution.

Concretely, **Environment-Regularized Policy Optimization (ERPO)** introduces a **Query-KL (QKL)** term that bounds this query distribution shift, together with a dataset-static reference-derived per-query weight that biases each per-query update toward queries typical under the reference. The QKL gradient flows strictly through the query likelihood; the response score function used by policy-gradient estimators does not appear in the QKL term, so QKL exerts no direct gradient pressure on the response distribution—exploration is preserved. ERPO plugs into GRPO/PPO/REINFORCE-style pipelines without additional forward passes. On six mathematical reasoning benchmarks, **ERPO replaces** the standard Policy-KL regularizer while achieving effective control over query distribution drift, delivering stronger accuracy and substantially more stable behavior under high-temperature decoding and long-horizon training.Our source code are available at https://github.com/alibaba/ERPO.

1 Introduction

Policy optimization (PO) methods have become the de facto recipe for post-training large language models (LLMs), spanning trust-region style updates (TRPO/PPO) and preference-based objectives (DPO) together with broader RLHF/RLAIF variants (Schulman et al., 2015b, 2017; Ouyang et al., 2022; Bai et al., 2022; Rafailov et al., 2023). Despite impressive progress in mathematical reasoning and beyond, practitioners still face a persistent dilemma: *how to trade off training stability against effective exploration*. In long-horizon runs, optimization noise and distribution shift tend to accumulate, leading to oscillations and occasional collapses.

We argue that a key and under-controlled source of instability is *environment non-stationarity* induced by the **query distribution**. Even with a fixed training corpus, the model's own sequence likelihood over training queries co-evolves with the policy: as θ updates, the likelihood the model assigns to each prompt drifts, altering the effective training environment and amplifying gradient variance. This input-side non-stationarity mirrors classic RL settings in which either the initial-state distribution or the transition kernel drifts over time; non-stationary and robust RL therefore advocate explicit distributional control (Padakandla, 2021; Iyengar, 2005; Nilim and El Ghaoui, 2005). A related lesson from imitation learning is that policy updates induce covariate (state) shift, motivating interactive data aggregation such as DAgger/AggreVaTe (Ross et al., 2011; Ross and Bagnell, 2014).

Recent LLM work formalizes prompt distributions (EVA, Align-Pro (Ye et al., 2024; Trivedi et al., 2025)) or reweights training data (StablePrompt, WPO (Kwon et al., 2024; Zhou et al., 2024)), while mainstream RLHF PO focuses on action-side Policy-KL to an SFT reference (Schulman et al., 2015b, 2017; Ouyang et al., 2022); neither directly constrains the query distribution. Empirically (Figure 1), under a fixed Policy-KL budget the batch-estimated Query-KL rises steadily while Policy-KL stays flat—the policy-induced query dis-

*Corresponding author.

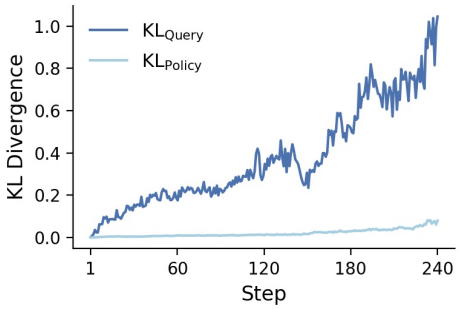

Figure 1: KL losses during GRPO training. The Query-KL (dark) rises while the Policy-KL (light) stays low, showing action-only KL does not stabilize the query process.

tribution ρ_θ drifts unchecked from its pre-RL reference ρ_{θ_0}.

We treat the model's query likelihood as the input-side statistic to regularize. We introduce **Query-KL regularization (QKL)**, a penalty on the KL divergence between the current policy-induced query distribution ρ_θ and a *pre-RL reference* ρ_{θ_0}, limiting inter-round drift of this likelihood while leaving the action space free to explore. In parallel, we propose a lightweight reference-derived per-query weight that biases each per-query update toward queries typical under ρ_{θ_0}, reducing estimator variance and improving robustness under high-temperature decoding—where LLMs are especially sensitive to the long tail of decoding distributions (Holtzman et al., 2020; Wang et al., 2023). Both components are estimator-agnostic and compatible with GRPO/PPO/REINFORCE-style implementations with minimal changes. Figure 2 sketches ERPO: on top of GRPO we replace the usual Policy-KL with a Query-KL term, and during advantage computation we reweight per-query contributions by a reference-derived prior, yielding an environment-aware update while preserving action-side exploration.

We make four main contributions. (1) **Query-environment control:** We treat the model's query likelihood as the regularizable input-side statistic, combining *Query-KL (QKL)* to bound its drift from a pre-RL reference ρ_{θ_0} with a dataset-static reference-derived per-query weight to reduce variance and tame high-temperature behavior. (2) **Estimator-agnostic instantiation:** The method adds a QKL term and a reference-derived per-query weight on top of GRPO/PPO/REINFORCE-style pipelines with minimal changes. (3) **Stability evaluation:** We assess RL stability via *multi-temperature* sampling paired with a *multi-metric* suite (Pass@k, Pass@1, Avg@k), enabling comprehensive capability and robustness evaluation. (4) **Empirical gains:** Across diverse reasoning benchmarks, the approach consistently improves accuracy.

2 Related Works

2.1 Reinforcement Learning with Verifiable Rewards (RLVR)

Reinforcement Learning with Verifiable Rewards represents a paradigm shift from traditional RLHF approaches by leveraging automatically verifiable outcomes rather than human preference annotations. This approach is particularly powerful for domains where ground truth can be objectively determined, such as mathematical reasoning, code generation, and logical problem solving. Models like AlphaCode (Li et al., 2022) and recent mathematical reasoning (Jeannotte and Kieran, 2017; Xia et al., 2025) systems leverage execution results and correctness verification as direct reward signals, eliminating the need for expensive human annotation.

Process Reward Models (PRMs) have emerged as a sophisticated extension of RLVR, where intermediate steps in reasoning processes are evaluated and rewarded based on their correctness (Uesato et al., 2022; Lightman et al., 2023). Recent developments include tool-augmented reasoning systems (Schick et al., 2023) and self-verification approaches (Kojima et al., 2022), which combine language models with external verification tools to enable automatic reward computation for broader task domains.

While RLVR provides scalable and consistent training signals compared to subjective human preferences, it introduces unique challenges in handling high variance from sparse rewards and potential reward hacking behaviors. These stability issues motivate the need for robust training methodologies that can effectively leverage verifiable rewards while maintaining training stability.

2.2 Reinforcement Learning Stability in Language Model Training

The stability of reinforcement learning algorithms in language model training has become a critical research area due to unique challenges posed by discrete action spaces, large parameter spaces, and complex reward landscapes (Sutton et al., 1998).

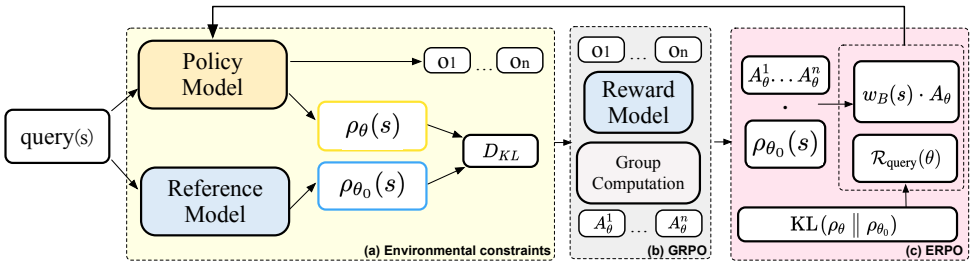

Figure 2: **The Proposed ERPO Overview.** (a) For each query, the policy and reference induce current and reference query samplers, and we pre-compute a Query-KL to penalize environment drift. (b) For each query, the policy samples a response group scored by the reward model to produce the standard GRPO learning signal. (c) On top of GRPO we replace response-KL with pre-computed Query-KL and weight within-query advantages by the query's occurrence probability, yielding an environment-aware update.

Recent works have identified specific stability issues including reward hacking (Gao et al., 2023) and the alignment tax problem (Dai et al., 2025), where policy optimization can degrade downstream performance while improving target metrics. Distribution shift during training has been recognized as a fundamental source of instability in policy gradient methods (Reddy et al., 2020). In language model contexts, this manifests as shifts in the query distribution during training, leading to high variance in gradient estimates and potential policy collapse (Wen et al., 2024). Existing approaches primarily focus on action-space regularization through trust region methods (Schulman et al., 2015a) and KL divergence penalties between current and reference policies.

Despite progress in understanding RL stability issues, there remains a notable gap in explicitly managing the input query distribution during training. Most current approaches focus on output regularization rather than addressing environmental shifts at the input level, leaving query distribution management as an underexplored avenue for improving training stability.

3 Preliminaries

3.1 RLVR setting and notation

We consider a standard generative–verify (RLVR) setting for training a large language model (LLM) with parameters θ. Given a query $q \in \mathcal{Q}$, the LLM defines a response policy $\pi_\theta(o \mid q)$ over $o \in \mathcal{O}$, and a verifier returns a scalar reward $g(q, o) \in \mathbb{R}$. The underlying RL objective is

$$J_{\mathrm{RL}}(\theta) = \mathbb{E}_{q\sim\rho_{train},\, o\sim\pi_\theta(\cdot|q)}\big[g(q,o)\big], \quad (1)$$

where ρ_{train} denotes the query distribution in the training set.

In practice, group-based or advantage-based variants of Eq. (1) are widely used (e.g., GRPO (Shao et al., 2024), RLOO (Ahmadian et al., 2024), DAPO (Yu et al., 2025)); our method is compatible with any such estimator. For concreteness in experiments we adopt the group-relative formulation of GRPO:

$$A_\theta^{\mathrm{GRPO}}(q, o^{(k)}) = \frac{g(q, o^{(k)}) - \mathrm{mean}(\mathbf{g})}{\mathrm{std}(\mathbf{g})}, \quad (2)$$

where $\mathbf{g} = \{g(q, o^{(k)})\}_{k=1}^K$ are the rewards of K responses sampled from $\pi_\theta(\cdot \mid q)$ for the same query.

3.2 Policy-induced query distribution and environment drift

During LLM-RL training, the queries seen at each step are jointly shaped by the training corpus, curriculum design, difficulty filters, and active sampling schedulers.

Rather than tying the analysis to such pipeline-specific factors, we introduce a *policy-only* notion: ρ_θ is the *policy-induced query distribution*, defined by $\rho_\theta(q) = P_\theta(q)$ — the model's autoregressive sequence likelihood of q treated as a self-terminating token sequence (see Section 4 for the autoregressive form). This model-induced distribution is distinct from the exogenous ρ_{train}, which supplies queries to training batches and remains fixed in our experiments.

By construction, ρ_θ depends only on θ, and the pre-RL reference ρ_{θ_0} inherits whatever data composition shaped π_{θ_0} (pretraining, SFT, or prior RL stages). As θ is updated, ρ_θ drifts away from ρ_{θ_0}; without explicit constraint, this drift is unchecked. We call this *policy-induced environment drift*, and

quantify it by the KL divergence

$$\begin{aligned}\mathrm{EnvShift}(\theta) &:= \mathrm{KL}(\rho_\theta \,\|\, \rho_{\theta_0}) \\ &= \mathbb{E}_{q\sim\rho_\theta}\Big[\log \tfrac{\rho_\theta(q)}{\rho_{\theta_0}(q)}\Big].\end{aligned} \tag{3}$$

Empirically (Figure 1), under a fixed action-level KL budget the batch-estimated Query-KL keeps rising throughout training while the response-level Policy-KL stays nearly flat—constraining only the action distribution does not stabilize the input/query process. This motivates the two components in Section 4: a query-level KL that bounds environment drift, and a per-query weighting scheme that stabilizes the empirical objective under arbitrary pipeline biases.

4 Method

We treat the pre-RL model-induced query distribution ρ_{θ_0} as the *reference environment*. As θ changes, ρ_θ may drift while the exogenous ρ_{train} remains fixed; our *Environment-Regularized Policy Optimization* (ERPO) bounds the former while leaving response-side exploration unconstrained.

Working assumption (A1). We treat as a premise—not a theorem—that maintaining alignment between the model-induced ρ_θ and its pre-RL reference ρ_{θ_0} better preserves the generalization capabilities inherited from prior training stages than allowing unconstrained drift. A1 is examined empirically in Section 5.

Query likelihood. Our method relies on the autoregressive sequence likelihood the model assigns to a query. For a query q tokenized as $(x_1, \ldots, x_T)$, write $P_\theta(q) \triangleq \prod_t P_\theta(x_t \mid x_{<t})$ for this sequence likelihood and $\ell_\theta(q) \triangleq \log P_\theta(q) = \sum_t \log P_\theta(x_t \mid x_{<t})$ for its log form, with $P_{\theta_0}(q)$ and $\ell_{\theta_0}(q)$ defined analogously under the reference.

The reference ℓ_{θ_0} is computed once over the training set and cached, while ℓ_θ is read off the per-step PG forward pass; the cached ℓ_{θ_0} table and the per-step ℓ_θ are the only inputs ERPO needs beyond what the underlying PG estimator already computes (see Appendix F for full computational notes).

4.1 Population objective and empirical loss

Let $\bar{g}_\theta(q) := \mathbb{E}_{o\sim\pi_\theta(\cdot|q)}[g(q,o)]$ be the per-query expected reward, so that $J_{\mathrm{RL}}(\theta) = \mathbb{E}_{q\sim\rho_{\text{train}}}[\bar{g}_\theta(q)]$. ERPO adds a query-level regularizer $\mathcal{R}_{\text{query}}(\theta)$ penalizing the divergence between the model-induced ρ_θ and ρ_{θ_0}:

$$J_{\mathrm{ERPO}}(\theta) := J_{\mathrm{RL}}(\theta) - \alpha\, \mathcal{R}_{\text{query}}(\theta), \tag{4}$$

where $\alpha > 0$ controls regularization strength. On a mini-batch $B = \{q_i\}_{i=1}^m$, we additionally reweight per-query contributions by a dataset-static, reference-derived weight $w_B(q)$ (Section 4.2); together with a batch-level KL estimate $\widehat{\mathcal{R}}_{\text{query}}(\theta)$, this gives the empirical loss

$$\begin{aligned}\widehat{L}_{\mathrm{ERPO}}(\theta) := &-\frac{1}{m}\sum_{q\in B} w_B(q)\,\bar{g}_\theta(q) \\ &+ \alpha\,\widehat{\mathcal{R}}_{\text{query}}(\theta).\end{aligned} \tag{5}$$

w_B enters only at the estimator level; it shapes the SGD update direction toward queries typical under ρ_{θ_0} but does not enter the target J_{ERPO}.

4.2 Query-level KL and query reweighting

Query-level KL. We instantiate the regularizer as the KL from the current query distribution to the reference:

$$\begin{aligned}\mathcal{R}_{\text{query}}(\theta) &:= \mathrm{KL}\big(\rho_\theta \,\big\|\, \rho_{\theta_0}\big) \\ &= \mathbb{E}_{q\sim\rho_\theta}\big[\log \rho_\theta(q) - \log \rho_{\theta_0}(q)\big].\end{aligned} \tag{6}$$

QKL *decouples regularization from exploration*: the penalty acts solely on the query distribution through $\ell_\theta(q)$, while imposing no direct constraint on $\pi_\theta(o \mid q)$. Conventional Policy-KL, in contrast, restricts response behavior and consumes the action-side exploration budget.

Proposition 1 (Structural Decoupling). *For autoregressive π_θ, the gradient of $\mathcal{R}_{\text{query}}(\theta)$ admits the closed form*

$$\begin{aligned}\nabla_\theta \mathcal{R}_{\text{query}}(\theta) = \mathbb{E}_{q\sim\rho_\theta}\Big[&\big(\ell_\theta(q) - \ell_{\theta_0}(q)\big) \\ &\cdot \nabla_\theta \ell_\theta(q)\Big],\end{aligned} \tag{7}$$

which flows strictly through $\nabla_\theta \ell_\theta(q)$; the response score function $\nabla_\theta \log \pi_\theta(o \mid q)$ used by PG estimators does not appear in the QKL loss, so the regularizer exerts no direct gradient pressure on the response distribution. Proof in Appendix H.

In our fixed-dataset implementation, we evaluate a K3-style mini-batch surrogate using the per-step $\ell_\theta(q)$ and cached $\ell_{\theta_0}(q)$. It regularizes model-induced query-likelihood drift without treating model likelihood as external query frequency. The resulting $\widehat{\mathcal{R}}_{\text{query}}(\theta)$ is used in Eq. (5).

Query reweighting. Instead of optimizing the standard RL objective in Eq. (1) under the empirical query distribution ρ_{train}, ERPO targets the reference-aligned query prior ρ_{θ_0}:

$$J_{\text{ERPO}}(\theta) = \mathbb{E}_{q\sim\rho_{\theta_0},\, o\sim\pi_\theta(\cdot|q)}[g(q,o)]. \quad (8)$$

Because stochastic training samples queries from ρ_{train}, this objective can be written as an importance-weighted objective with ideal weight

$$w^\star(q) = \frac{\rho_{\theta_0}(q)}{\rho_{\text{train}}(q)}. \quad (9)$$

Under approximately uniform sampling from a dataset of size N, $w^\star(q) = N\rho_{\theta_0}(q)$, so the query weight only needs to preserve the relative ordering of the cached reference-induced query prior. We therefore use a bounded, dataset-static weight $w(q) \propto \ell_{\theta_0}(q)$, leading to

$$L_{\text{ERPO}}(\theta) = -\mathbb{E}_{\substack{q\sim\rho_{\text{train}} \\ o\sim\pi_\theta(\cdot|q)}}[w(q)g(q,o)] + \alpha\mathcal{R}_{\text{query}}(\theta). \quad (10)$$

The derivation is given in Appendix I.

PG-compatible surrogate. ERPO is agnostic to the inner PG estimator: any surrogate of the form $\nabla_\theta \bar{g}_\theta(q) \approx \mathbb{E}_o[u_\theta(q,o)A^\star_\theta(q,o)\nabla_\theta \log \pi_\theta(o \mid q)]$ recovers GRPO ($u_\theta \equiv 1$, $A^\star$ from Eq. (2)), PPO (clipped ratios), or REINFORCE (sample reward) by appropriate choice of $u_\theta, A^\star_\theta$. Plugging into Eq. (5) gives the ERPO mini-batch PG loss

$$\mathcal{L}_{\text{PG}}(\theta) := -\frac{1}{m}\sum_{q\in B}\frac{w_B(q)}{K}\sum_{o\in\mathcal{G}(q)} u_\theta(q,o)\, A^\star_\theta(q,o) + \alpha\,\widehat{\mathcal{R}}_{\text{query}}(\theta), \quad (11)$$

where $\mathcal{G}(q)$ is the K-response group sampled for q. Per-algorithm specializations (GRPO/PPO/REINFORCE) and the resulting loss decompositions are summarized in Appendix G.

5 Experiments

5.1 Experimental Setup

Training We conduct experiments on mathematical reasoning tasks using Level 3–5 problems from the MATH dataset (Hendrycks et al., 2021), totaling approximately 8.5K examples. These are used to evaluate our proposed ERPO method, in comparison with the vanilla GRPO baseline. As described in Appendix A, the model must wrap its intermediate reasoning in `<think></think>` tags, and place the final answer inside `\boxed {}`.

Evaluation We follow standard practice and assess performance on six widely used benchmarks: AIME24, AIME25, AMC, MATH500 (Hendrycks et al., 2021), Minerva (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). Prior work typically reports Avg@K (Yu et al., 2025), Pass@1 (Liu et al., 2025), and Pass@K (Hao et al., 2025) after RLVR training, often without specifying or controlling the inference-time sampling temperature. This omission can substantially affect reported performance and render results across studies not directly comparable. In preliminary experiments, we found that inference-time sampling temperature has a significant impact on performance, and that the effect intensifies as training progresses. To control for this factor, we fix the number of training steps across all models and evaluate at temperatures from 0.1 to 1.5; performance is then aggregated over this range.

Implementation Details We conduct all experiments using the EasyR1 framework (Zheng et al., 2025), training the Qwen2.5-Math-7B and Qwen2.5-32B model (Yang et al., 2024; Qwen et al., 2025) with both GRPO and ERPO algorithms. Following prior work (Liu et al., 2025), we set the maximum sequence length to 3K tokens. For each problem, we sample eight responses at an inference temperature of 1.0. The rollout batch size is set to 512, and the update batch size to 128, for a total of 240 training steps. Token-level loss is applied throughout training. To ensure a fair comparison, we adopt the default KL divergence coefficient of 0.01.

5.2 Main Results

Figure 3 summarizes Avg@32 accuracy on six mathematical reasoning benchmarks, averaged over sampling temperatures from 0.1 to 1.5. ERPO consistently outperforms GRPO, with gains of up to 14.9% and an overall average improvement of 6.2%, highlighting its enhanced capability. Table 1 presents the detailed results for each benchmark, grouped by evaluation metric (e.g., Pass@1, Pass@K).

For both GRPO and ERPO, the prompts are identical to those used during training, whereas the Qwen base model adopts the default configuration from Dr.GRPO (Liu et al., 2025) to ensure optimal performance. Consistent with the aggregated results in Figure 3 and Table 1, ERPO surpasses GRPO across all evaluation metrics, achiev-

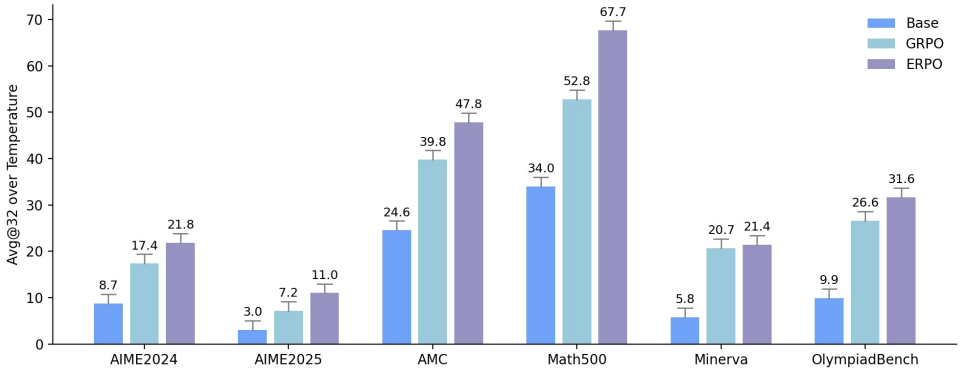

Figure 3: Avg@32 over Sampling Temperatures on Mathematical Reasoning Tasks

Table 1: Performance comparison across mathematical reasoning benchmarks. Best results per column are highlighted in bold.

Method	*AIME24*	*AIME25*	*AMC*	*MATH500*	*Minerva*	*Olympiad*	Avg.
Mean Avg@32							
Base	0.087	0.030	0.246	0.340	0.058	0.099	0.143
GRPO	0.174	0.072	0.398	0.528	0.207	0.266	0.274
ERPO	**0.218**	**0.110**	**0.478**	**0.677**	**0.214**	**0.316**	**0.336**
Mean Pass@32							
Base	0.373	0.206	0.674	0.764	0.349	0.411	0.463
GRPO	0.471	0.287	0.768	0.850	**0.516**	0.558	0.575
ERPO	**0.509**	**0.342**	**0.820**	**0.904**	0.500	**0.593**	**0.611**
Mean Pass@1							
Base	0.090	0.038	0.264	0.342	0.062	0.099	0.149
GRPO	0.169	0.084	0.398	0.533	0.201	0.263	0.275
ERPO	**0.207**	**0.091**	**0.477**	**0.679**	**0.217**	**0.320**	**0.332**

ing improvements of 6.2% in Avg@32, 3.64% in Pass@32, and 5.69% in Pass@1. We also applied the concept of ERPO to other RLVR algorithms and observed similarly effective gains; details are provided in Appendix E.

5.3 Training Dynamics

Figure 4 illustrates the training dynamics of the ERPO method. For both approaches, the sampling accuracy on the training set remains largely consistent; however, their divergence from the reference model exhibits markedly different trajectories.

In GRPO, constraints are imposed on the action distribution, causing the query distribution to drift away from the reference model at a substantially faster rate. Consequently, the KL divergence at the query level is an order of magnitude greater than at the policy level.

This imbalance leads to pronounced discrepancies in performance between the training and evaluation datasets. In contrast, ERPO applies constraints directly to the query distribution and adjusts the loss according to the probability of the given problem. This design both limits the degree of divergence from the reference model during training and, by leveraging the independence between the problem and the response, allows unconstrained exploration at the policy level. A quantitative analysis can be found in Appendix C. As a result, ERPO achieves superior generalization performance on general problems.

To assess the stability of long-term RL training, we scale the training steps up to 1K and monitor changes in model performance over time. As shown in the figure 5 and 6, GRPO remains stable for sampling temperatures below 1.0 until approximately 240 steps (epoch=15). However, a pronounced performance degradation is first observed in the high-temperature sampling regime after 400 steps, and subsequently propagates to encompass

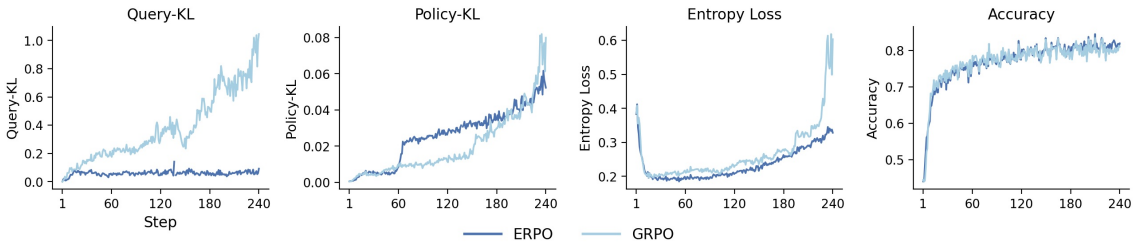

Figure 4: Training Dynamics on ERPO

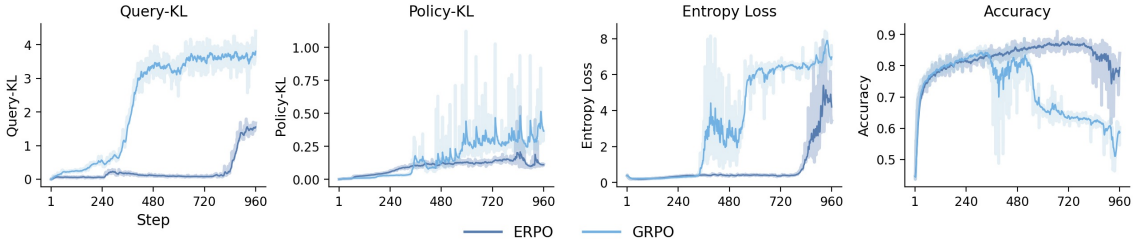

Figure 5: Training Dynamics on Long-term RL

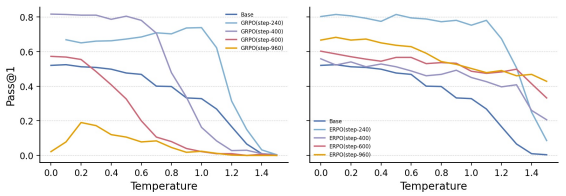

Figure 6: Performance Variation Across Training Steps

sampling across all temperatures as the steps increase.

In contrast, ERPO exhibits a modest performance decline; however, the overall deterioration is substantially smaller, and its performance even improves within the high-temperature range. Figure 6 presents the complete training trajectories for both GRPO and ERPO. Although ERPO is not entirely immune to the collapse phenomenon that may occur during extended training—manifested as a sudden increase in entropy and a loss of sampling capability—it consistently outperforms vanilla GRPO and achieves a comparable degree of policy distribution constraint without relying on an explicit policy-based KL divergence term.

5.4 Analysis

Ablation Study We conduct ablation studies on the MATH500 benchmarks to assess reasoning efficiency. Table 2 summarizes the results for several commonly used sampling temperatures. Figure 7 further provides the complete performance–temperature variation curves across different experimental settings, along with the corresponding training dynamics.

Mechanisms Without modifying other hyperparameters, replacing the policy-based KL divergence with query-based KL divergence yields the best overall performance[1], with an average improvement of 15.9% over GRPO. In contrast, GRPO with policy-based KL divergence shows its highest performance only at a temperature of 1.0 (see Figure 7(a)).

To further investigate, an ablation study is conducted on the two mechanisms of ERPO with their effects evaluated using KL divergence and entropy (Table 3 and Figure 7(d)). The term $w_{B(s)}$ down-weights gradients from low-probability queries, which often lead to low-probability responses, thereby increasing gradient variance and entropy (Quantitative analysis is provided in Appendix D). Introducing $w_{B(s)}$ allows sufficient training while concurrently reducing the policy KL divergence.

Different regularization strengths α also exert a significant influence on performance. As the constraint strength increases (e.g., $\alpha = 5 \times 10^{-2}$), the model achieves further improvements in overall performance (see Table 2). It is worth noting that we did not conduct an exhaustive search for the optimal α; instead, we retained the default value to ensure a relatively fair comparison.

[1]We also experimented with completely removing all KL divergence constraints, which resulted in the training process failing to converge. Therefore, we only report GRPO results under the default-weight KL divergence constraint.

Table 2: Performance Comparison Under Different Experimental Settings

Base Model	Method	α	$w(s)$	Rollout Count	Temperature Metrics					
					0.1	**0.6**	**1**	**1.5**	≤ 1.0	**1.2–1.5**
Baseline	—	—	—	—	52.40	46.80	32.80	0.40	44.44	6.15
Qwen-7B	GRPO	1×10^{-2}	—	8	66.80	68.40	73.80	0.40	68.80	12.50
	GRPO*	1×10^{-2}	✓	8	78.20	76.80	71.20	7.60	76.14	23.79
	ERPO	1×10^{-2}	—	8	**81.60**	**81.60**	**79.00**	2.60	**80.90**	38.00
	ERPO	1×10^{-2}	✓	8	79.40	80.60	75.20	8.60	78.74	37.90
		5×10^{-3}	✓	8	53.80	60.60	66.20	**15.40**	59.94	39.30
		5×10^{-2}	✓	8	78.80	81.00	76.00	15.00	79.00	**43.35**
Qwen-7B under $n = 16$	GRPO	1×10^{-2}	—	16	73.00	79.20	75.00	10.60	75.22	39.75
	ERPO	1×10^{-2}	✓	16	80.40	78.80	74.40	56.20	77.82	66.25
Qwen-32B	GRPO	1×10^{-2}	—	8	81.60	82.40	81.20	25.20	81.62	57.20
	ERPO	1×10^{-2}	✓	8	85.00	84.80	83.60	80.80	84.60	82.80

Note: Within the Qwen-7B ($n = 8$) setting, the best result in each column is highlighted in **bold**, while the second-best result is underlined. No highlighting is applied to the Qwen-7B ($n = 16$) or Qwen-32B results. The first three Qwen-7B rows constitute the ablation study, comparing GRPO, GRPO*, and ERPO without query reweighting. The following three rows constitute the hyperparameter study, evaluating ERPO with query reweighting under different values of α. The columns ≤ 1.0 and **1.2–1.5** report the mean accuracy (Acc) over the corresponding temperature ranges. The $w(s)$ column indicates whether query reweighting is applied (✓) or not (—). GRPO* denotes GRPO using only query reweighting.

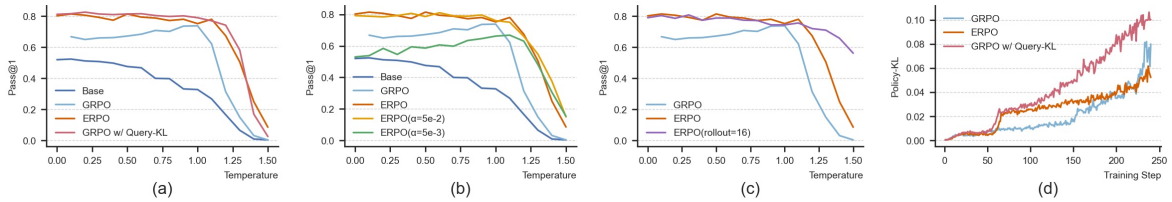

Figure 7: Pass@1 accuracy and training dynamics under different settings: (a)–(c) Model performance at various temperatures on MATH500; (d) Policy-KL divergence variation with GRPO using only Query-KL.

Table 3: Influence of Query-KL and Query-Reweighting on Training Stability

Method	Query-KL	Policy-KL	Entropy
GRPO (Avg@3)	0.9679	0.0601	0.5063
GRPO w/$w_{B(s)}$	0.5933	**0.0113**	**0.2782**
GRPO w/Query-KL	**0.0041**	0.1001	0.5674
ERPO (Avg@3)	0.0828	0.0728	0.4244

Rollouts We also analyze the effect of the number of samples per query. By increasing the sampling number to 16, we achieve the best performance, with the average Pass@1 rising to 74.6%. A higher sampling count also significantly improves sampling stability at high temperatures (see Table 2), without a noticeable increase in divergence from the reference model. Moreover, increasing the sampling count facilitates ERPO-based models in acquiring the correct reasoning format more effectively. [2]

6 Conclusion

By analyzing the coupling between the environment and the policy space in large language models, we decouple parameter regularization from the optimization objective during training. Specifically, we employ query-level KL divergence to bound the drift of the policy-induced query distribution ρ_θ from a pre-RL reference ρ_{θ_0}. We further reweight the advantage by a dataset-static reference-derived per-query weight, biasing updates toward queries typical under ρ_{θ_0} and preventing premature convergence to suboptimal solutions. Experiments across multiple mathematical reasoning benchmarks demonstrate that the proposed ERPO method can achieve comparable

[2] Across multiple experiments, the GRPO method consistently failed to capture the desired output format. Consequently, for all experiments, we report only the answer accuracy.

KL divergence control without explicit policy regularization, while delivering superior performance. Furthermore, by sampling at different temperatures, we examine the evolution of sampling capability over long-term RL training, providing additional evidence of ERPO's stability during training.

Limitations

Our experiments focus primarily on mathematical reasoning benchmarks and Qwen-family models, so the extent to which ERPO transfers to broader instruction-following, dialogue, code-generation, and multilingual settings remains to be validated. ERPO also relies on estimating query-level likelihoods or prevalence statistics during training; the quality and computational cost of these estimates may vary with the data-selection mechanism and model scale. Finally, we did not conduct an exhaustive sweep over the regularization coefficient, leaving more systematic hyperparameter analysis for future work.

References

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. 2024. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. *Preprint*, arXiv:2402.14740.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, and 12 others. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Juntao Dai, Taiye Chen, Yaodong Yang, Qian Zheng, and Gang Pan. 2025. Mitigating reward overoptimization in rlhf via behavior-supported regularization. *arXiv preprint arXiv:2503.18130*.

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR.

Yaru Hao, Li Dong, Xun Wu, Shaohan Huang, Zewen Chi, and Furu Wei. 2025. On-policy rl with optimal reward baseline. *arXiv preprint arXiv:2505.23585*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, and 1 others. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Math. *Cornell University - arXiv,Cornell University - arXiv*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*.

Garud N. Iyengar. 2005. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280.

Doris Jeannotte and Carolyn Kieran. 2017. A conceptual model of mathematical reasoning for school mathematics. *Educational Studies in mathematics*, 96(1):1–16.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Minchan Kwon, Gaeun Kim, Jongsuk Kim, Haeil Lee, and Junmo Kim. 2024. Stableprompt: Automatic prompt tuning using reinforcement learning for large language model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9868–9884, Miami, Florida, USA. Association for Computational Linguistics.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, R'emi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, and 1 others. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Arnab Nilim and Laurent El Ghaoui. 2005. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Sindhu Padakandla. 2021. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys*, 54(6):127:1–127:25.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Siddharth Reddy, Anca Dragan, Sergey Levine, Shane Legg, and Jan Leike. 2020. Learning human objectives by evaluating hypothetical behavior. In *International conference on machine learning*, pages 8020–8029. PMLR.

Stephane Ross and J. Andrew Bagnell. 2014. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA. PMLR.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015a. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR.

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

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Richard S Sutton, Andrew G Barto, and 1 others. 1998. *Reinforcement learning: An introduction*. MIT press Cambridge.

Prashant Trivedi, Souradip Chakraborty, Avinash Reddy, Vaneet Aggarwal, Amrit Singh Bedi, and George K. Atia. 2025. Align-pro: A principled approach to prompt optimization for llm alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*.

Jiaxin Wen, Ruiqi Zhong, Akbir Khan, Ethan Perez, Jacob Steinhardt, Minlie Huang, Samuel R Bowman, He He, and Shi Feng. 2024. Language models learn to mislead humans via rlhf. *arXiv preprint arXiv:2409.12822*.

Shijie Xia, Xuefeng Li, Yixin Liu, Tongshuang Wu, and Pengfei Liu. 2025. Evaluating mathematical reasoning beyond accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 27723–27730.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, and 1 others. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Feng Yao, Liyuan Liu, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. 2025. Your efficient rl framework secretly brings you off-policy rl training.

Ziyu Ye, Rishabh Agarwal, Tianqi Liu, Rishabh Joshi, Sarmishta Velury, Quoc V. Le, Qijun Tan, and Yuan Liu. 2024. Scalable reinforcement post-training beyond static human prompts: Evolving alignment via asymmetric self-play. *arXiv preprint arXiv:2411.00062*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yaowei Zheng, Junting Lu, Shenzhi Wang, Zhangchi Feng, Dongdong Kuang, and Yuwen Xiong. 2025. Easyr1: An efficient, scalable, multi-modality rl training framework. https://github.com/hiyouga/EasyR1.

Wenxuan Zhou, Ravi Agrawal, Shujian Zhang, Sathish Reddy Indurthi, Sanqiang Zhao, Kaiqiang Song, Silei Xu, and Chenguang Zhu. 2024. Wpo: Enhancing rlhf with weighted preference optimization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8328–8340, Miami, Florida, USA. Association for Computational Linguistics.

A Prompt

```
{{ content | trim }} You FIRST think about the reasoning process as an internal monologue and then provide the final answer. The reasoning process MUST BE enclosed within <think> </think> tags. The final answer MUST BE put in \boxed{}.
```

B Variation of Metrics with Temperature

Figure 8 illustrates the model performance across different evaluation metrics and sampling temperatures. Our approach reduces the performance gap between different sampling temperatures, while increasing the likelihood of sampling correct outputs.

C Analysis of Reward Hacking

In the course of our experiments, we observed a severe reward hacking phenomenon when training with the baseline GRPO method. Specifically, while the model consistently achieved high rewards on the training data, its performance on the evaluation set often plateaued or even degraded during the later stages of optimization. This pronounced discrepancy suggests that the model overfits to the specific characteristics of the reward signal during training sampling, failing to generalize to the standard decoding distribution used during inference.

To quantitatively investigate this issue, we monitored the Train–Evaluation Consistency throughout the training process. We periodically evaluated both the training accuracy and the evaluation accuracy every ten optimization steps. To ensure the robustness of our inference metrics, evaluation was conducted using vLLM under two different Tensor Parallelism settings (TP1 and TP2), a factor which has been shown in previous work (Yao et al., 2025) to impact model performance.

Table 4 summarizes the average performance gap across six key checkpoints (Steps 40, 80, 120, 160, 200, and 240). The results confirm our hypothesis:

- GRPO exhibits a substantial average gap of 6.47%, indicating a significant misalignment between training and inference behaviors. Notably, as shown in the detailed trajectories in Table 5, GRPO's evaluation accuracy drops sharply at Step-240 (from $\approx 75\%$ to 58.4%) despite maintaining high training accuracy, a classic signature of reward hacking.
- ERPO, in contrast, demonstrates superior consistency. It reduces the average Train–Eval gap by approximately 51% (from 6.47% to 3.14%).

This significant reduction in the performance gap indicates that ERPO effectively regularizes the training process, preventing the model from exploiting spurious patterns in the reward function and ensuring that improvements in training translate reliably to inference performance.

D Correlation analysis between query and response probabilities

We sampled over 8K questions from the training dataset at a temperature of 1.0 and independently computed the negative log-likelihood (NLL) for both the prompt and response parts:

$$\mathrm{NLL}(x) = -\sum \log p(x)$$

where x denotes the generation probabilities of tokens in the prompt or response. The resulting histogram is shown in Figure 9. We observe a positive correlation between the NLL of the prompt and that of the response. For 95% of the training samples (where the NLL of the prompt is less than 300), the correlation coefficient is close to 1. Low-probability responses appearing in positive samples contribute to an increase in entropy during training.

E ERPO On Different Algorithms

Additional experiments were conducted on DAPO(Yu et al., 2025) and RLOO(Ahmadian et al., 2024) with and without the global KL divergence constraint, yielding absolute improvements of 10.24% and 2.28% at temperatures below 1.0, respectively (see Table 6). These findings demonstrate that the proposed method can achieve significant gains when applied to other RLVR algorithms.

F Query Likelihood and the Cached Reference

This appendix expands the query likelihood machinery and computational notes referenced from Section 4.

Autoregressive sequence likelihood. For a query q with tokenization $x = (x_1, \ldots, x_T)$, the model's sequence log-likelihood under parameters

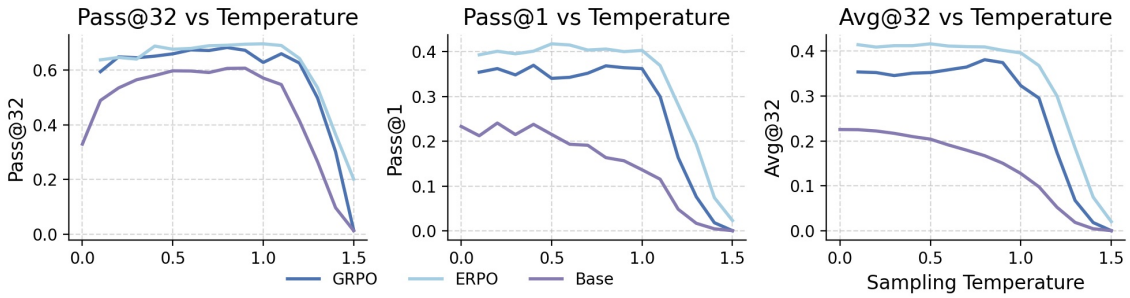

Figure 8: Variation of Metrics with Temperature

Table 4: **Quantification of Reward Hacking via Train–Inference Gap.** The table compares the average accuracy during training sampling versus inference decoding. A larger gap indicates severe reward hacking (overfitting to training dynamics). ERPO reduces this gap by $\approx 51\%$, demonstrating robust generalization.

Method	**Avg. Train Acc**	**Avg. Eval@TP1**	**Avg. Eval@TP2**	**Gap@TP1**	**Gap@TP2**	**Avg. Gap**
GRPO	77.3	69.8	70.1	7.5	5.5	6.47
ERPO	77.1	74.1	73.8	3.0	3.3	3.14
Improvement	–	–	–	↓ 4.5 (60%)	↓ 2.2 (40%)	↓ 3.33 (51%)

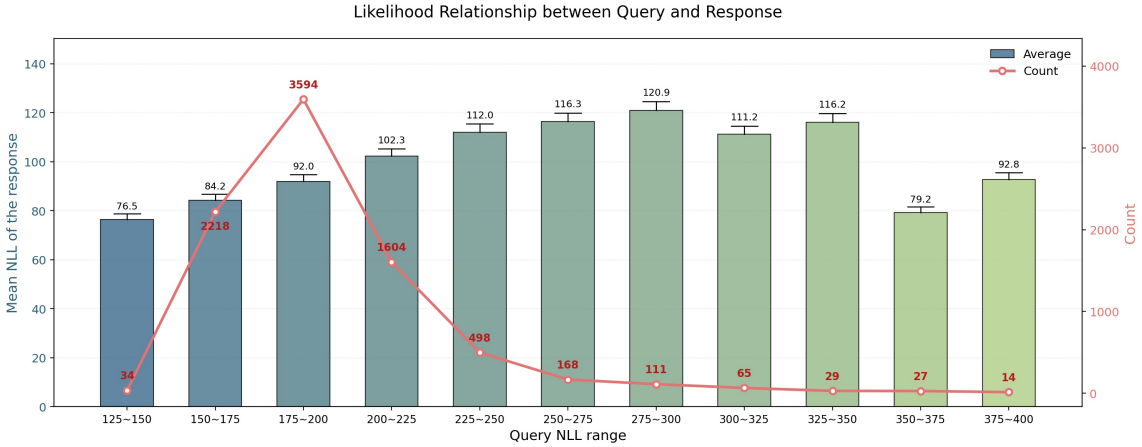

Figure 9: Likelihood Relationship between Query and Response

θ is

$$\ell_\theta(q) \triangleq \log P_\theta(x) = \sum_{t=1}^{T} \log P_\theta(x_t \mid x_{<t}). \tag{12}$$

This is well-defined for any token sequence, regardless of whether q is sampled on-policy from ρ_θ, drawn from a fixed dataset, or written by a human; computing $\ell_\theta(q)$ requires a single forward pass of the LLM. The reference $\ell_{\theta_0}(q)$ is defined analogously under π_{θ_0}.

Computational notes governing ERPO's zero-overhead design. Two distinctions in how ℓ_θ and ℓ_{θ_0} are obtained make ERPO essentially free relative to the underlying PG estimator:

- $\ell_{\theta_0}(q)$ is computed once over the entire training set using the reference model π_{θ_0}, prior to RL, and cached. The per-query table is reused throughout training and does not change. Under backpropagation it is treated as a constant.
- $\ell_\theta(q)$ is computed per training step using the current θ. This forward pass is already performed by the underlying PG estimator (for per-query advantage evaluation), so QKL and QW reuse it at no additional forward cost.

The cached ℓ_{θ_0} table and the per-step ℓ_θ are the only inputs ERPO needs beyond what the underlying PG estimator already computes. In particular, the query weight $w_B(q)$ from Eq. (9) is fully precomputed and contributes no gradient or extra forward pass; only the QKL estimator $\widehat{\mathcal{R}}_{\text{query}}(\theta)$ uses the per-step $\ell_\theta(q)$, and even there the value is read from

Table 5: **Trajectory of Train vs. Inference Accuracy.** Detailed performance recorded at 40-step intervals. Note the divergence in GRPO at Step-240, where Eval accuracy drops significantly while Train accuracy remains high—a clear sign of reward hacking. ERPO maintains consistency throughout.

Model / Metric	**Step-0**	**Step-40**	**Step-80**	**Step-120**	**Step-160**	**Step-200**	**Step-240**	**Avg.**
GRPO (Baseline)								
Train Acc	44.48	76.41	76.09	78.29	78.95	79.44	76.70	72.91
Eval@TP1	31.20	73.20	76.00	73.40	72.20	73.60	58.40	65.43
Eval@TP2	30.60	74.00	74.80	74.20	76.60	75.80	66.20	67.46
ERPO (Ours)								
Train Acc	44.55	75.63	76.53	78.66	80.63	78.41	81.46	73.70
Eval@TP1	31.20	74.00	77.20	77.00	78.40	78.60	78.40	70.69
Eval@TP2	30.60	72.60	77.00	76.60	77.40	78.60	80.20	70.43

Table 6: Performance Comparison Under Different Experimental Settings

Base Model	**Method**	**D_KL**	**Rollout**	**Temperature**					
				0.1	**0.6**	**1**	**1.5**	**≤1.0**	**1.2-1.5**
Baseline	–	–	–	52.4	46.8	32.8	0.4	44.44	6.15
Qwen-7B	GRPO	Policy	8	66.8	68.4	73.8	0.4	68.8	12.5
			16	73.0	79.2	75.0	10.6	75.22	39.75
	ERPO	Query	8	79.4	80.6	75.2	8.6	78.74 (+9.94)	37.9
			16	80.4	78.8	74.4	<u>56.2</u>	77.82 (+2.6)	<u>66.25</u>
	DAPO	–	8	62.0	77.4	65.4	5.6	68.16	14.0
	DAPO+ERPO	Query		80.2	79.4	75.8	20.2	78.4 (+10.24)	36.93
	RLOO	Policy	8	77.6	78.4	75.4	12.4	77.28	35.93
	RLOO+ERPO	Query		81.2	80.4	79.4	17.6	79.56 (+2.28)	40.8
Qwen-32B	GRPO	Policy	8	<u>81.6</u>	<u>82.4</u>	<u>81.2</u>	25.2	<u>81.62</u>	57.2
	ERPO	Query		**85.0**	**84.8**	**83.6**	**80.8**	**84.6** (+2.98)	**82.8**

Note: The columns **≤1.0** and **1.2–1.5** show the mean accuracy (Acc) over the corresponding temperature ranges. Values in parentheses indicate improvements over baseline methods. Best results per column are highlighted in **bold**, second-best results are <u>underlined</u>.

the existing PG forward pass.

G PG-Compatible Surrogate: Per-Algorithm Specializations

The ERPO mini-batch loss $\mathcal{L}_{\mathrm{PG}}(\theta)$ in Eq. (11) is instantiated by a particular choice of action-level weight $u_\theta(q,o)$ and advantage $A^\star_\theta(q,o)$, summarized below. The two ERPO modifications—adding the query-level KL term and reweighting the outer per-query sum by $w_B(q)$—are orthogonal to the choice of $(u_\theta, A^\star_\theta)$ and act only on the outer query loop.

GRPO. Set $u_\theta \equiv 1$ and $A^\star_\theta = A^{\mathrm{GRPO}}_\theta$ from Eq. (2). Substituting yields

$$\mathcal{L}_{\text{ERPO-GRPO}}(\theta) := -\frac{1}{m}\sum_{q\in B}\frac{w_B(q)}{K}\sum_{o\in\mathcal{G}(q)} A^{\mathrm{GRPO}}_\theta(q,o)\,\log\pi_\theta(o\mid q) + \alpha\,\widehat{\mathcal{R}}_{\mathrm{query}}(\theta). \tag{13}$$

All GRPO engineering details (reward normalization, group size K, sampling temperature, etc.) remain unchanged.

PPO. We set

$$u_\theta(q,o) = \mathrm{clip}\left(\frac{\pi_\theta(o\mid q)}{\pi_{\theta_{\mathrm{old}}}(o\mid q)},\,1-\epsilon,\,1+\epsilon\right)$$

and A^*_θ to the (GAE-based or standard) PPO advantage. The inner sum recovers the PPO clipped

surrogate, with the outer $w_B(q)$ and the added $\alpha\,\widehat{\mathcal{R}}_{\text{query}}$ providing the ERPO wrapping.

REINFORCE. Set $u_\theta \equiv 1$ and $A^\star_\theta(q,o) = g(q,o) - b_\theta(q)$ for an arbitrary baseline $b_\theta(q)$; the inner sum recovers the sample-wise REINFORCE estimator.

Drop-in modification recipe. Given an existing PG-style training pipeline using any of the above estimators, applying ERPO requires three steps: (i) precompute ℓ_{θ_0} once over the dataset (Appendix F); (ii) replace the per-query outer weight $1/m$ with $w_B(q)/m$ from Eq. (9); (iii) add $\alpha\,\widehat{\mathcal{R}}_{\text{query}}(\theta)$ (K3-estimated from the per-step ℓ_θ and the cached ℓ_{θ_0}) to the loss. No additional forward passes, no architectural changes, and no modification to the inner PG estimator's clip/baseline logic are required.

H Proof of Proposition 1: Structural Decoupling of Regularization and Exploration

This appendix provides the formal proof of Proposition 1 stated in Section 4.2.

Proposition 1 (restated). *For the autoregressive generation process parameterized by θ, the gradient of the Query-KL regularizer $\mathcal{R}_{\text{query}}(\theta)$ strictly flows through the query log-likelihood $\nabla_\theta \ell_\theta(q)$ and is entirely orthogonal to the response policy score function $\nabla_\theta \log \pi_\theta(o \mid q)$.*

Proof. By definition,

$$\mathcal{R}_{\text{query}}(\theta) = \sum_q \rho_\theta(q)\big(\log \rho_\theta(q) - \log \rho_{\theta_0}(q)\big).$$

Differentiating with respect to θ and applying the product rule,

$$\begin{aligned}\nabla_\theta \mathcal{R}_{\text{query}}(\theta) &= \sum_q \nabla_\theta \rho_\theta(q)\big(\log \rho_\theta(q) \\ &\qquad - \log \rho_{\theta_0}(q)\big) \\ &\quad + \sum_q \rho_\theta(q)\,\nabla_\theta\big(\log \rho_\theta(q) \\ &\qquad - \log \rho_{\theta_0}(q)\big).\end{aligned} \tag{14}$$

Second sum collapses. Since ρ_{θ_0} does not depend on θ, we have $\nabla_\theta \log \rho_{\theta_0}(q) = 0$. Applying the identity $\nabla_\theta \log \rho_\theta(q) = \nabla_\theta \rho_\theta(q)/\rho_\theta(q)$ to the remaining term gives

$$\begin{aligned}\sum_q \rho_\theta(q)\,\nabla_\theta \log \rho_\theta(q) &= \sum_q \nabla_\theta \rho_\theta(q) \\ &= \nabla_\theta\left(\sum_q \rho_\theta(q)\right) \\ &= \nabla_\theta(1) = 0,\end{aligned} \tag{15}$$

where the third equality uses the fact that ρ_θ is a probability distribution and therefore sums to one identically in θ.

First sum simplifies via the log-derivative trick. Applying $\nabla_\theta \rho_\theta(q) = \rho_\theta(q)\,\nabla_\theta \log \rho_\theta(q)$ and writing $\ell_\theta(q) := \log \rho_\theta(q)$ and $\ell_{\theta_0}(q) := \log \rho_{\theta_0}(q)$, we obtain the closed-form gradient

$$\begin{aligned}&\nabla_\theta \mathcal{R}_{\text{query}}(\theta) \\ &\quad = \mathbb{E}_{q\sim\rho_\theta}\Big[\big(\ell_\theta(q) - \ell_{\theta_0}(q)\big) \\ &\qquad \cdot \nabla_\theta \ell_\theta(q)\Big].\end{aligned} \tag{16}$$

Structural conclusion. The right-hand side of Eq. (16) depends on θ only through the query log-likelihood $\ell_\theta(q)$ (and its gradient $\nabla_\theta \ell_\theta(q)$); no factor of the response policy score function $\nabla_\theta \log \pi_\theta(o \mid q)$ appears. Equivalently, perturbations to $\pi_\theta(o \mid q)$ that leave the query marginal ρ_θ unchanged incur zero QKL gradient and are unconstrained by $\mathcal{R}_{\text{query}}$. This establishes the structural decoupling between input-environment regularization (carried by QKL) and response-side exploration (left untouched). □

I Derivation of Query Reweighting

The standard RL objective optimizes the expected reward under the empirical training query distribution ρ_{train}. ERPO instead considers a reference-aligned objective in which queries are sampled from the cached reference-induced query prior ρ_{θ_0}:

$$J_{\text{ERPO}}(\theta) = \mathbb{E}_{q\sim\rho_{\theta_0},\, o\sim\pi_\theta(\cdot|q)}\big[g(q,o)\big]. \tag{17}$$

Expanding the expectation gives

$$J_{\text{ERPO}}(\theta) = \sum_q \rho_{\theta_0}(q) \sum_o \pi_\theta(o \mid q) g(q,o). \tag{18}$$

However, stochastic training samples queries from the empirical distribution ρ_{train}, rather than directly from ρ_{θ_0}. To express the same objective under

ρ_{train}, we multiply and divide by $\rho_{\text{train}}(q)$:

$$J_{\text{ERPO}}(\theta) = \sum_q \rho_{\text{train}}(q) \frac{\rho_{\theta_0}(q)}{\rho_{\text{train}}(q)} \sum_o \pi_\theta(o \mid q) g(q,o)$$
$$= \mathbb{E}_{q\sim\rho_{\text{train}},\, o\sim\pi_\theta(\cdot|q)} \left[w^\star(q) g(q,o) \right], \tag{19}$$

where the ideal importance weight is

$$w^\star(q) = \frac{\rho_{\theta_0}(q)}{\rho_{\text{train}}(q)}. \tag{20}$$

If training queries are sampled approximately uniformly from a dataset of size N, then $\rho_{\text{train}}(q) = 1/N$. Therefore,

$$w^\star(q) = \frac{\rho_{\theta_0}(q)}{\rho_{\text{train}}(q)} = N\rho_{\theta_0}(q). \tag{21}$$

Since the multiplicative constant N is shared by all queries, it does not affect the relative weighting among queries. Thus, up to a global constant,

$$w^\star(q) \propto \rho_{\theta_0}(q). \tag{22}$$

In practice, ERPO estimates the reference-induced query prior using a cached reference score $\ell_{\theta_0}(q)$, which assigns larger values to queries that are more favored under the pretrained reference model. Hence we use a bounded, dataset-static query weight $w(q)$ that is monotone in $\ell_{\theta_0}(q)$:

$$w(q) \propto \ell_{\theta_0}(q). \tag{23}$$

Substituting this query weight into the negative optimization objective yields the ERPO training loss

$$L_{\text{ERPO}}(\theta) = -\mathbb{E}_{q\sim\rho_{\text{train}},\, o\sim\pi_\theta(\cdot|q)} \left[w(q) g(q,o) \right]$$
$$+ \alpha \mathcal{R}_{\text{query}}(\theta). \tag{24}$$

where $\mathcal{R}_{\text{query}}(\theta)$ denotes the query-level regularization term and α controls its strength.

In practice, the exact reference-induced query prior $\rho_{\theta_0}(q)$ is not directly available on a finite training set. Therefore, we use the cached reference-model log-probability as a practical monotone proxy for the relative query preference induced by the reference model. This approximation only aims to preserve the relative ordering of queries under the reference model, rather than to estimate the absolute density value. We emphasize that this implementation is not intended as an unbiased density-ratio estimator; rather, it is a bounded monotone approximation to the ideal query importance weight in Eq. (20).

For each query q_i, let $\log p_{\theta_0}(q_i)$ denote its cached sequence log-probability under the reference model. We first define its negative log-probability as

$$s_i = -\log p_{\theta_0}(q_i). \tag{25}$$

A larger reference-model probability corresponds to a smaller s_i. To obtain a weight whose magnitude increases with the reference-model query probability, we use the inverse normalized negative log-probability. For a dataset with N queries, let

$$\bar{s} = \frac{1}{N} \sum_{j=1}^{N} s_j. \tag{26}$$

We define the unbounded query weight as

$$\tilde{w}(q_i) = \frac{\bar{s}}{s_i}. \tag{27}$$

This normalization makes the weight dimensionless and centers its scale around the dataset average, while preserving the monotonic relationship that higher-probability queries receive larger weights. Finally, to reduce the variance of importance reweighting and prevent any single query from dominating optimization, we clip the weight to a fixed range:

$$w(q_i) = \text{clip}\left(\tilde{w}(q_i), 0, 2\right). \tag{28}$$