RETHINKING KL REGULARIZATION IN RLHF: FROM WRONG VALUE ESTIMATION TO CORRECT GRADIENT OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement Learning from Human Feedback (RLHF) often uses a Kullback–Leibler (KL) penalty to stabilize training and control drift from a reference policy. However, recent implementations sometimes choose KL estimator formulas based on Monte Carlo value-estimation criteria such as bias and variance, and then reuse these formulas with gradient-carrying policy parameters as optimization losses. We show that this transfer from value estimation to loss design is wrong in general: good KL value estimation does not imply a correct KL gradient. We introduce a gradient-centric framework that compares KL implementations by the scalar coefficient they induce on the policy score function, covering both detached k_n **in reward** terms and directly differentiated k_n **as loss** terms. Through this framework, we first prove that k_1 **as loss**, although an unbiased KL value estimator, has zero expected on-policy gradient and provides no reference-directed regularization. We then prove that the PPO-style k_1 **in reward** implementation and the decoupled k_2 **as loss** implementation are gradient-equivalent and recover the Reverse KL (RKL) gradient under on-policy sampling. This equivalence, first proven here, identifies both as theoretically sound implementations of the RKL objective. In contrast, the GRPO-style k_3 **as loss** objective induces a biased first-order approximation to the principled RKL coefficient, with mismatched tail behavior. We further show that off-policy "as loss" KL terms require explicit importance-sampling corrections. Controlled experiments validate these findings, providing a comprehensive, gradient-based rationale for choosing and correctly implementing KL regularization in RLHF.

1 INTRODUCTION

Training state-of-the-art Large Language Models (LLMs) is a multi-stage process. Following large-scale pretraining and Supervised Fine-Tuning (SFT), a final post-training stage, Reinforcement Learning from Human Feedback (RLHF), is often employed to align models with preferences and to improve performance on demanding tasks such as mathematical reasoning (Ouyang et al., 2022; Shao et al., 2024). A core component of RLHF is **KL regularization**: a penalty derived from the Kullback–Leibler divergence (Kullback & Leibler, 1951) that constrains the policy to stay close to a reference model (typically the SFT checkpoint), thereby improving training stability and reducing reward overfitting (Ouyang et al., 2022; Stiennon et al., 2020).

Despite the critical role of the KL loss, its theoretical foundations in optimization remain under-explored. Across algorithms and codebases, the KL penalty takes several subtly different forms, and a common justification is how well the resulting Monte Carlo estimator estimates the KL *value* (e.g., bias/variance). Yet the same estimator formula is often reused with gradient-carrying policy parameters as a differentiable training loss. In RLHF, this distinction matters: because the sampling distribution depends on the policy parameters, good value estimation does not guarantee that the induced gradient matches the intended regularizer. As a result, implementations that look nearly identical in code—often differing only in whether a term is detached or backpropagated—can correspond to qualitatively different update directions and stability behavior.

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.

This paper argues that a gradient-centric perspective is essential for designing robust and effective RLHF algorithms. We establish a unified framework that connects two common implementation styles: using the term k_n as a detached coefficient (**k_n in reward**) versus differentiating a surrogate penalty directly (**k_n as loss**). This framework allows us to compare implementations through the scalar coefficient they induce on the score function $\nabla_\theta \log \pi_\theta$.

Our main contributions are threefold:

1. **A gradient-correctness criterion for KL loss design.** We first prove that conclusions from KL value estimation do not necessarily translate into correct optimization behavior. In particular, **k_1 as loss** provides a clean counterexample: despite k_1 being an unbiased KL value estimator, it has zero expected gradient under on-policy sampling and thus provides no KL constraint.
2. **Identifying principled RKL implementations.** We derive the exact on-policy Reverse KL (RKL) gradient and prove that **k_1 in reward** (PPO-style) and **k_2 as loss** are gradient-equivalent, establishing both as theoretically sound choices for RLHF regularization.
3. **Diagnosing GRPO-style surrogates and off-policy pitfalls.** We analyze **k_3 as loss** (used in GRPO) as a biased first-order surrogate with mismatched tail behavior. We further identify missing importance-sampling and clipping corrections as a key source of bias when "as loss" heads are reused off-policy, and provide a principled correction.

2 Related Work

Our work intersects three lines of research: KL value estimation, RLHF systems, and KL loss design. We briefly review each and clarify how our gradient-centric perspective complements prior contributions.

KL Value Estimation. The KL divergence is often estimated via Monte Carlo sampling, motivating a family of single-sample scalar terms k_1, k_2, k_3 (defined in Section 3) (John, 2020). Prior discussions evaluate them as *value estimators*, characterizing k_1 as unbiased but high-variance, k_2 as biased but lower-variance, and k_3 as an "optimal" low-variance unbiased estimator under regularity assumptions. We complement this view in two ways: Section I shows that the advertised superiority of k_3 can fail under distribution shift and heavy-tailed importance ratios; and more importantly, we argue that value-estimation criteria are insufficient when these same scalar terms are used as *optimization losses*, where the induced gradient is what determines the regularization effect.

RLHF Systems and Methods. Open-source RLHF frameworks (e.g., OpenRLHF (Hu et al., 2024), verl (Sheng et al., 2024), slime (Zhu et al., 2025), and ROLL (Wang et al., 2025)) support PPO-style updates (Ouyang et al., 2022) and more recent critic-free variants. They emphasize systems-level challenges such as efficient rollouts (e.g., OpenRLHF uses vLLM (Kwon et al., 2023)) and actor–critic stability. Several recent works focus on the critic bottleneck: VAPO (Yue et al., 2025) proposes critic pretraining, while GRPO and REINFORCE++ remove the critic altogether to enable larger actor scaling. Many methods retain explicit KL regularization, although some rule-based reward algorithms, such as DAPO (Yu et al., 2025), remove it for performance; ProRL (Liu et al., 2025b) instead periodically resets the reference model, illustrating that drift-control mechanisms remain an important design choice. Our contribution is complementary: we analyze the gradient correctness of the KL regularizer itself, which affects stability regardless of the surrounding RLHF loss.

KL Loss in RLHF. The practice of adding a KL penalty *in the reward* traces back to InstructGPT (Ouyang et al., 2022), where the log-ratio acts as a detached coefficient multiplying the policy score function. Jaques et al. (2019) noted a potential connection between adding KL to the reward versus to the loss, but without a formal gradient-level analysis. More recently, GRPO (Shao et al., 2024) (adopted in DeepSeek-R1 (Guo et al., 2025)) popularized directly differentiating through a k_3-style surrogate, often justified by its value-estimation properties (John, 2020). Our work provides a unified framework to compare these choices at the gradient level, identifies when they match the intended Reverse KL objective, and discusses connections to Forward KL in Appendix L.

3 PRELIMINARIES

Before analyzing KL implementations, we introduce the single-sample scalar terms that are commonly discussed as KL value estimators but are also reused as KL regularizers or optimization losses, together with the standard RLHF objective they are meant to regularize.

3.1 SINGLE-SAMPLE KL TERMS

The Kullback–Leibler (KL) divergence from a distribution $q(x)$ to a reference $p(x)$ is defined as

$$D_{\mathrm{KL}}(q \parallel p) = \mathbb{E}_{x\sim q}\left[\log \frac{q(x)}{p(x)}\right].$$

When this expectation is intractable, practitioners estimate it from Monte Carlo samples. Defining the likelihood ratio $\delta(x) = p(x)/q(x)$, common single-sample terms include (John, 2020):

$$k_1(x) = -\log \delta(x), \quad k_2(x) = \tfrac{1}{2}\left(\log \delta(x)\right)^2, \quad k_3(x) = \delta(x) - 1 - \log \delta(x).$$

In RLHF, we instantiate these terms with $q(\cdot) = \pi_\theta(\cdot|x)$ (the current policy) and $p(\cdot) = \pi_{\mathrm{ref}}(\cdot|x)$ (the reference). The term k_3 is designed to be nonnegative and low-variance when p and q are close. However, Section I shows that this value-estimation property relies on regularity conditions: under support or tail mismatch, rare events can dominate its sample mean and its variance may become extremely large or infinite. More importantly for this paper, the same scalar formulas can also be used inside policy optimization—for example as detached reward coefficients or as directly differentiated losses. Once used for *regularization*, their suitability is determined by the gradient they induce, not only by their scalar value-estimation properties.

Clarification. The blog (John, 2020) cited by GRPO to justify k_3 discusses only *value estimation*—not *loss design*. Even as a value estimator, Section I shows that k_3 can fail under distribution shift.

3.2 THE RLHF OBJECTIVE

RLHF fine-tunes a policy $\pi_{\boldsymbol{\theta}}$ to produce responses y to prompts x that maximize a learned reward $r(x, y)$. Pure reward maximization can lead to reward hacking and drift from a trusted SFT policy. A standard remedy is to regularize toward a fixed reference π_{ref} via the Reverse KL (RKL):

$$\mathcal{J}_{\mathrm{RLHF}}(\boldsymbol{\theta}) = \mathbb{E}_{x,y\sim\pi_{\boldsymbol{\theta}}}\Big[r(x,y)\Big] - \beta\,\mathbb{E}_{x\sim\mathcal{D}}\Big[D_{\mathrm{KL}}\big(\pi_{\boldsymbol{\theta}}(\cdot|x) \parallel \pi_{\mathrm{ref}}(\cdot|x)\big)\Big] \triangleq \mathcal{J}_{\mathrm{reward}}(\boldsymbol{\theta}) - \beta\,\mathcal{J}_{\mathrm{RKL}}(\boldsymbol{\theta}). \tag{1}$$

Here $\mathcal{D}$ is the prompt distribution, $\beta > 0$ controls the strength of regularization, and in practice y is sampled from a detached snapshot π_θ that is numerically equal to $\pi_{\boldsymbol{\theta}}$ at sampling time.

The high-level objective is clear, but it leaves open an important implementation choice: *how should the KL term be integrated into the policy-gradient update?* Different choices lead to different induced gradients. We address this question next.

4 A UNIFIED FRAMEWORK FOR KL REGULARIZATION

Although most RLHF algorithms optimize the same high-level objective (Equation (1)), their concrete implementations of the KL term differ. This section introduces a unified framework that makes these differences explicit and comparable.

The key observation is that any practical KL implementation, regardless of its form, ultimately induces a scalar coefficient $c(x, y)$ multiplying the policy score function $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)$. This coefficient determines both the expected update direction and the variance of the stochastic gradient. By making $c(x, y)$ explicit, we can directly compare implementations: a KL loss is *gradient-correct* if and only if its induced coefficient matches that of the target KL objective.

Key convention (red vs. black). We use $\boldsymbol{\theta}$ (red) to denote trainable parameters that carry gradients, while θ (black) denotes a detached snapshot. Samples y are drawn from $\pi_\theta(\cdot|x)$, which is numerically equal to $\pi_{\boldsymbol{\theta}}(\cdot|x)$ at sampling time. Scalar coefficients multiplying the score function are always treated as detached.[1]

4.1 Core Components of the RLHF Objective

The practical RLHF objective consists of two terms—reward maximization and KL regularization—both estimated via Monte Carlo sampling.

Reward Maximization. The primary goal is to maximize expected reward: $\mathcal{J}_{\text{reward}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\boldsymbol{\theta}}(\cdot|x)}[r(x,y)]$. In the discrete autoregressive LLM setting considered here, sampled responses are not reparameterized, so the gradient is estimated via the log-derivative trick (REINFORCE (Williams, 1992); see Appendix B):

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\text{reward}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\boldsymbol{\theta}}(\cdot|x)}\left[r(x,y)\cdot\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y|x)\right].$$

Here $r(x,y)$ is typically a shaped advantage signal rather than a raw reward (details in Appendix A).

KL Regularization. The RKL term constrains the policy toward the reference:

$$\mathcal{J}_{\text{RKL}}(\boldsymbol{\theta}) := \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\boldsymbol{\theta}}(\cdot|x)}\left[\log\frac{\pi_{\boldsymbol{\theta}}(y|x)}{\pi_{\text{ref}}(y|x)}\right]. \tag{2}$$

Like the reward term, practical implementations estimate this expectation with samples $y\sim\pi_\theta(\cdot|x)$ and differentiate only through $\pi_{\boldsymbol{\theta}}$ (see Appendix C). However, *how* the KL-derived term interacts with the gradient varies across algorithms. We distinguish two common styles:

1. k_n **in reward (detached coefficient):** The KL-derived scalar k_n multiplies the score function as a *detached* coefficient—gradients do not flow through k_n itself. This is the style used in PPO, with $k_1(y|x) = \log\pi_\theta(y|x) - \log\pi_{\text{ref}}(y|x)$:
$$\mathcal{J}_{k_n\text{ in reward}}(\boldsymbol{\theta}) := \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\boldsymbol{\theta}}(\cdot|x)}\left[k_n\big(\pi_\theta(y|x),\pi_{\text{ref}}(y|x)\big)\log\pi_{\boldsymbol{\theta}}(y|x)\right].$$
2. k_n **as loss (direct differentiation):** The KL-derived term is treated as a standalone loss, with gradients propagated *through* it:
$$k_3(y|x) = \frac{\pi_{\text{ref}}(y|x)}{\pi_{\boldsymbol{\theta}}(y|x)} - \log\frac{\pi_{\text{ref}}(y|x)}{\pi_{\boldsymbol{\theta}}(y|x)} - 1 \quad \text{(GRPO)},$$
$$k_2(y|x) = \frac{1}{2}\left(\log\frac{\pi_{\boldsymbol{\theta}}(y|x)}{\pi_{\text{ref}}(y|x)}\right)^2 \quad \text{(Ours)}.$$
The corresponding objective is:
$$\mathcal{J}_{k_n\text{ as loss}}(\boldsymbol{\theta}) := \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\boldsymbol{\theta}}(\cdot|x)}\left[k_n\big(\pi_{\boldsymbol{\theta}}(y|x),\,\pi_{\text{ref}}(y|x)\big)\right].$$

Both forms are *optimization surrogates*: their scalar values need not equal a KL divergence; what matters is the gradient they induce.

Connecting the two styles. Although k_n as loss differentiates through k_n directly, its gradient can always be written in the $k_{n'}$ in reward form for some equivalent coefficient $k_{n'}$. This coefficient is obtained by differentiating k_n with respect to the log-policy:

Equivalent coefficient formula:

$$k_{n'}\big(\pi_\theta(y|x),\,\pi_{\text{ref}}(y|x)\big) := \frac{\partial\, k_n\big(\pi_{\boldsymbol{\theta}}(y|x),\,\pi_{\text{ref}}(y|x)\big)}{\partial\,\log\pi_{\boldsymbol{\theta}}(y|x)}\Big|_{\pi_{\boldsymbol{\theta}}=\pi_\theta}.$$

This formula maps an k_n as loss head to its equivalent detached score-function coefficient. It does not imply that every detached k_n in reward coefficient comes from a useful direct loss. A key example is $k_{2'} = k_1$: in Section 5.2, we prove that k_1 in reward and k_2 as loss are therefore gradient-equivalent.

[1] We treat the entire response y as a single action, working with the joint probability $\pi(y|x)$ rather than token-level factors. This bandit-style simplification suffices for analyzing the core gradient properties.

4.2 Integration Forms and Algorithm Mapping

The KL formulation also determines how it is integrated with the reward objective in practice.

Combined vs. Decoupled Forms. Since k_n in reward shares the same score function as the reward term, its coefficient can be merged to form a single "advantage":

$$\mathcal{L}_{\text{combined}}(\boldsymbol{\theta}) := -\mathbb{E}_{x,y}\big[A_{\text{combined}}(x,y)\log \pi_{\boldsymbol{\theta}}(y|x)\big], A_{\text{combined}}(x,y) := r(x,y) - \beta k_n\big(\pi_\theta(y|x), \pi_{\text{ref}}(y|x)\big).$$

In contrast, k_n as loss naturally yields a **decoupled** objective with a separate KL head:

$$\mathcal{L}_{\text{decoupled}}(\boldsymbol{\theta}) := -\mathbb{E}_{x,y}[r(x,y)\log \pi_{\boldsymbol{\theta}}(y|x)] + \beta\, \mathbb{E}_{x,y}\left[k_n\left(\pi_{\boldsymbol{\theta}}(y|x), \pi_{\text{ref}}(y|x)\right)\right].$$

Off-policy considerations. In off-policy updates (e.g., PPO with multiple gradient steps per rollout), the combined form inherits importance-sampling (IS) and clipping corrections via the standard PPO surrogate $\pi_{\boldsymbol{\theta}}/\pi_{\theta_k}$. In contrast, k_n as loss implementations must apply IS and clipping to the KL head itself—a step commonly omitted in practice, leading to biased gradients (see Section G).

Positioning PPO and GRPO. Table 1 summarizes where common algorithms fall in this taxonomy. PPO-style KL shaping and KL-regularized REINFORCE++ variants use $\boldsymbol{k_1}$ **in reward**, typically in a combined form; GRPO uses $\boldsymbol{k_3}$ **as loss** in a decoupled form.

Table 1: Taxonomy of RLHF KL implementations. The "style" column specifies whether the KL term appears as a detached score-function coefficient or as a directly differentiated loss; the "form" column specifies whether this KL contribution is merged into the advantage/reward head or kept decoupled. k_n in reward can also be used in a decoupled form.

Algorithm	Typical k_n	Style	Form	Off-Policy Notes
PPO / REINFORCE++	k_1	in reward	Combined: $A_{\text{combined}} = r - \beta k_1$	IS/clipping inherited from PPO-style surrogate.
GRPO	k_3	as loss	Decoupled: reward head $+\beta k_3$ loss	Requires explicit IS/clipping; often omitted.

With the framework in place, we now analyze which KL losses are gradient-correct for the RKL objective and diagnose failure modes when they are not.

5 Gradient-Based Analysis of KL Implementations

We now apply the coefficient framework to analyze common KL implementations. The analysis proceeds in three steps: (i) a counterexample showing that $\boldsymbol{k_1}$ **as loss** provides no regularization signal; (ii) derivation of the principled RKL gradient and its equivalent surrogates; and (iii) diagnosis of $\boldsymbol{k_3}$ **as loss** as a biased first-order approximation.

5.1 Counterexample: Why $\boldsymbol{k_1}$ as loss Fails

We begin with a counterexample that illustrates the gap between value estimation and optimization. Although $k_1(y|x) = \log(\pi_\theta(y|x)/\pi_{\text{ref}}(y|x))$ is an unbiased estimator of the KL value, using it directly as a loss provides no regularization.

Consider the direct-loss formulation with on-policy sampling:

$$\mathcal{J}_{k_1 \text{ as loss}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[\log \pi_{\boldsymbol{\theta}}(y|x) - \log \pi_{\text{ref}}(y|x)\Big].$$

Since π_{ref} does not depend on $\boldsymbol{\theta}$, differentiating yields

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_{k_1 \text{ as loss}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\Big].$$

The gradient is independent of π_{ref}—it carries *no* directional signal toward the reference. Moreover, by the zero-mean score identity (Lemma C.2), this expectation equals zero. In finite-sample (minibatch) practice, the gradient is nonzero but has zero mean; $\boldsymbol{k_1}$ **as loss** thus injects reference-independent noise that increases variance and can destabilize training, without providing any meaningful constraint.

This counterexample underscores a key point: good value estimation does not imply good optimization. A KL regularizer must be evaluated by its induced gradient, not its scalar estimate.

5.2 PRINCIPLED RKL SURROGATES: k_1 IN REWARD $\Leftrightarrow$ k_2 AS LOSS

Having seen a failure mode, we now derive the gradient that a principled KL regularizer *should* produce. Applying the product rule and log-derivative trick to the RKL objective (Equation (2); see Appendix C) gives:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_{\mathrm{RKL}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D}}\left[\sum_y \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(y|x)\left(\log \frac{\pi_\theta(y|x)}{\pi_{\mathrm{ref}}(y|x)} + 1\right)\right].$$

By the zero-mean score identity in Lemma C.2, the term '+1' vanishes in expectation, resulting in the practical form of the policy gradient:

Target RKL Gradient:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_{\mathrm{RKL}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[\underbrace{\big(\log \pi_\theta(y|x) - \log \pi_{\mathrm{ref}}(y|x)\big)}_{\text{coefficient } k_1(y|x)} \underbrace{\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)}_{\text{score function}}\Big]. \tag{3}$$

Any surrogate that correctly implements RKL must reproduce this target gradient. The following theorem shows that two seemingly different designs achieve this.

Theorem 5.1 (Gradient equivalence of principled RKL surrogates)**.** *Under on-policy sampling $y \sim \pi_\theta(\cdot|x)$, the following two objectives are **gradient-equivalent**—both produce Equation (3):*

$$\mathcal{J}_{k_1 \text{ in reward}} = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[\underbrace{\big(\log \pi_\theta(y|x) - \log \pi_{\mathrm{ref}}(y|x)\big)}_{\text{coefficient } k_1 \text{ (no grad)}} \cdot \log \pi_{\boldsymbol{\theta}}(y|x)\Big] \tag{4}$$

$$\mathcal{J}_{k_2 \text{ as loss}} = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[\tfrac{1}{2}\underbrace{\big(\log \pi_{\boldsymbol{\theta}}(y|x) - \log \pi_{\mathrm{ref}}(y|x)\big)^2}_{\text{gradients flow through}}\Big] \tag{5}$$

Key insight. In Equation (4), k_1 is a fixed scalar—gradients flow only through $\log \pi_{\boldsymbol{\theta}}$. In Equation (5), gradients flow through the squared term. Yet differentiating the square produces exactly k_1 as the coefficient:

$$\nabla_{\boldsymbol{\theta}} \frac{1}{2}\left(\log \frac{\pi_{\boldsymbol{\theta}}(y|x)}{\pi_{\mathrm{ref}}(y|x)}\right)^2\Bigg|_{\pi_{\boldsymbol{\theta}}=\pi_\theta} = \underbrace{\left(\log \frac{\pi_\theta(y|x)}{\pi_{\mathrm{ref}}(y|x)}\right)}_{\text{same } k_1!} \cdot \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x).$$

Thus, both forms induce the same update: $k_1 \times$ score function. See Appendix C for the full proof.

Why use k_2 as loss? The squared form $k_2(y|x) = \frac{1}{2}(\log(\pi_\theta(y|x)/\pi_{\mathrm{ref}}(y|x)))^2$ is not theoretically superior to the gradient-equivalent **k_1 in reward**; both implement the same on-policy RKL gradient. Its practical appeal is that it provides a standalone differentiable KL head whose derivative automatically yields the required detached coefficient k_1. In contrast, implementing **k_1 in reward** requires explicitly treating k_1 as a detached score-function coefficient. The squared head is also non-negative as a scalar diagnostic and, unlike the k_3 surrogate, its induced coefficient does not contain the ratio $\pi_{\mathrm{ref}}(y|x)/\pi_\theta(y|x)$ whose variance scales with the chi-squared divergence. Thus, **k_2 as loss** is a convenient exact decoupled implementation of RKL, while **k_1 in reward** remains the equally principled PPO-style implementation. Off-policy updates require additional IS and clipping corrections; see Section G.

5.3 THE GRPO SURROGATE: k_3 as loss AS A FIRST-ORDER APPROXIMATION

We now turn to **k_3 as loss**, popularized by GRPO:

$$\mathcal{J}_{k_3 \text{ as loss}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[\frac{\pi_{\mathrm{ref}}(y|x)}{\pi_{\boldsymbol{\theta}}(y|x)} - \log \frac{\pi_{\mathrm{ref}}(y|x)}{\pi_{\boldsymbol{\theta}}(y|x)} - 1\right].$$

Differentiating yields the induced coefficient $k_{3'} = 1 - \pi_{\mathrm{ref}}(y|x)/\pi_\theta(y|x)$:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_{k_3 \text{ as loss}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[\left(1 - \frac{\pi_{\mathrm{ref}}(y|x)}{\pi_\theta(y|x)}\right)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\right] = \nabla_{\boldsymbol{\theta}} \mathcal{J}_{k_{3'} \text{ in reward}}(\boldsymbol{\theta}). \tag{6}$$

Table 2: KL regularization selection guide. RKL-correct methods induce $c^\star = k_1$; $\delta = \pi_{\text{ref}}/\pi_\theta$.

Implementation	Coeff.	RKL?	Off-policy	Recommendation
In-reward form: $\mathcal{L} = \mathbb{E}[k_n \cdot \log \pi_{\boldsymbol{\theta}}]$	(coefficient k_n does not carry gradients)			
$\boldsymbol{k_1}$ **in reward**	k_1	✓	Inherits PPO clip	**Recommended.** Exact RKL coefficient.
k_2 in reward	k_2	✗	–	Avoid for RKL. Mismatched coefficient.
k_3 in reward	k_3	✗	–	Avoid. Mixed RKL−FKL.
MiniMax-01	$\pi_\theta - \pi_{\text{ref}}$	–	Inherits PPO clip	Bounded $[-1, 1]$; MSE-based.
As-loss form: $\mathcal{L} = \mathbb{E}[k_n(\pi_{\boldsymbol{\theta}}, \pi_{\text{ref}})]$	(gradients flow through k_n)			
k_1 as loss	1	✗	–	**Avoid.** Zero expected gradient.
$\boldsymbol{k_2}$ **as loss**	k_1	✓	Needs IS/clip†	**Recommended.** Exact decoupled RKL.
k_3 as loss	$1-\delta$	≈	Needs IS/clip†	Biased 1st-order; mismatched tails.
DeepSeek-V3.2	$\rho \cdot k_1$	✓*	Built-in IS	Equivalent to k_1 in reward; unclipped can be unstable.

✓ = RKL-correct, ✗ = incorrect, ≈ = 1st-order approx. *Requires ρ to carry gradients. † See Section G.

Let $\delta(y|x) = \pi_{\text{ref}}(y|x)/\pi_\theta(y|x)$. The principled RKL coefficient and the k_3-induced coefficient are:

$$c^\star = -\log \delta, \qquad c_{3'} = 1 - \delta.$$

Around $\delta = 1$, Taylor expansion gives $-\log \delta \approx 1 - \delta + \mathcal{O}((\delta - 1)^2)$, so $c_{3'}$ is a first-order approximation of $c^\star$. The mismatch beyond first order leads to three issues. See Appendix E for formal statements and Figure 1 for visualization.

1. **Bias.** For $\delta \neq 1$, the update direction differs from the true RKL gradient.
2. **Mismatched tails.** When $\pi_\theta > \pi_{\text{ref}}$ ($\delta \to 0$), the principled coefficient $-\log \delta \to +\infty$ provides a strong restoring force. In contrast, $1 - \delta \to 1$ saturates, weakening the constraint precisely when drift is large. When $\pi_\theta < \pi_{\text{ref}}$ ($\delta \to \infty$), $1 - \delta \to -\infty$ diverges faster than $-\log \delta$, inducing potentially unbounded updates.
3. **Variance sensitivity.** The induced coefficient has variance
$$\text{Var}[1 - \delta] = \chi^2(\pi_{\text{ref}} \| \pi_\theta),$$
the chi-squared divergence, which can be unstable under distribution shift.

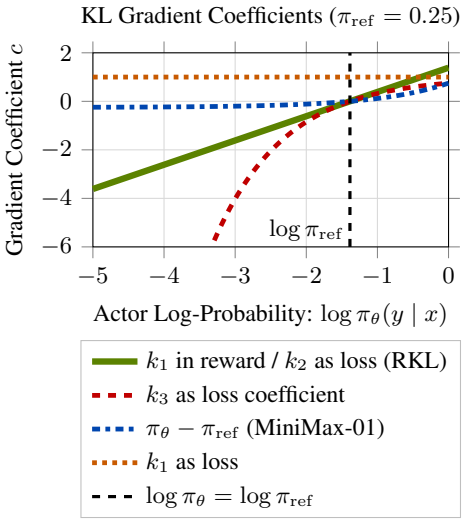

Figure 1: Gradient coefficient c induced on $\nabla_\theta \log \pi_\theta$ as a function of $\log \pi_\theta(y|x)$ for $\pi_{\text{ref}} = 0.25$. Principled RKL implementations yield $c = k_1$; k_3 as loss yields the first-order surrogate $c = 1 - \delta$; direct k_1 as loss yields $c = 1$.

Support mismatch can affect Monte Carlo KL estimation in general. The $\boldsymbol{k_3}$ **as loss** issue identified here is more specific: the same ratio $\pi_{\text{ref}}(y|x)/\pi_\theta(y|x)$ enters the gradient coefficient, so value-estimation variance becomes an optimization-level tail pathology. Appendix L provides an alternative view: $c_{3'} = 1 - \delta$ equals the Forward KL coefficient $-\delta$ plus an implicit baseline, making $\boldsymbol{k_3}$ **as loss** a variance-reduced FKL estimator rather than an RKL surrogate.

5.4 Summary of Recommendations

Table 2 summarizes the key practical consequences of our analysis. The "Coefficient" column shows the scalar c multiplying the score function $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}$; a method is RKL-correct if $c = k_1$. Detailed practical guidance is provided in Appendix K.

6 Experiments

We test our gradient analysis with controlled mathematical-reasoning RLHF experiments. They ask whether $\boldsymbol{k_1}$ **as loss** regularizes beyond noise and whether principled $\boldsymbol{k_2}$ **as loss** differs from first-

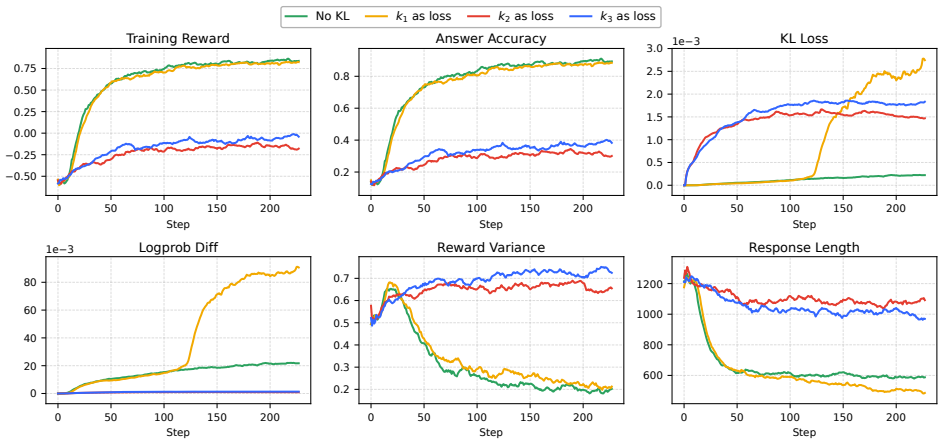

Figure 2: Main 7B training curves at $\beta = 0.5$. $\boldsymbol{k_1}$ **as loss** tracks the no-KL reward/score behavior but develops larger KL and log-probability gaps; $\boldsymbol{k_2}$ **as loss** implements the RKL-correct decoupled term, while $\boldsymbol{k_3}$ **as loss** has asymmetric tail behavior.

order $\boldsymbol{k_3}$ **as loss** as predicted by induced coefficients. We report the main 7B-scale Qwen2.5-Math curves here; Appendix N gives 1.5B dynamics, Appendix O reports downstream benchmarks, and Appendix P validates standard Qwen2.5-7B/32B checkpoints.

6.1 Setup

Dataset. We use a curated subset of OpenR1-Math-220k comprising NuminaMath 1.5 prompts with reference solutions verified by Math-Verify. After removing sequences exceeding 2048 tokens, 7,300 prompts remain; URLs and implementation details are provided in Appendix A.

Training Configuration. To isolate gradient effects, we use fully on-policy training: rollout batch size 32, 8 responses per prompt, update batch size 256, and sampling temperature 1.0. The actor is Qwen2.5-Math-7B (Yang et al., 2024); rewards combine regex-based format checking and Math-Verify accuracy. We disable entropy regularization. Unless otherwise stated, we set the KL weight to $\beta = 0.5$; to probe a weaker practical KL regime, we additionally report training metrics at $\beta = 0.001$ (which permits larger drift and makes instabilities easier to observe). A numerical stability issue with group normalization and our mitigation are discussed in Appendix J.

6.2 Results

We track reward, accuracy, KL/log-probability gaps to the reference, reward variance, and response length. Figure 2 presents a six-panel summary of the main 7B training metrics.

Experiment 1: $\boldsymbol{k_1}$ as loss vs. no KL. Figure 2 compares $\boldsymbol{k_1}$ **as loss** to a no-KL baseline. Since its gradient is independent of π_{ref} and has zero expectation, $\boldsymbol{k_1}$ **as loss** should behave like no KL plus noise. The two runs reach comparable reward/score, but $\boldsymbol{k_1}$ **as loss** exhibits much larger late KL and log-probability gaps, consistent with added stochasticity rather than reference-directed regularization.

Experiment 2: $\boldsymbol{k_2}$ as loss vs. $\boldsymbol{k_3}$ as loss. Figure 2 also compares the RKL-correct $\boldsymbol{k_2}$ **as loss** to the GRPO-style $\boldsymbol{k_3}$ **as loss**. The key issue with $\boldsymbol{k_3}$ **as loss** is its asymmetric tail behavior: its coefficient saturates in the high-policy-mass direction, yielding a weak restoring force, while the opposite tail can produce very large loss values.

Weak-KL stress test. Under a weaker KL weight ($\beta = 0.001$), $\boldsymbol{k_3}$ **as loss** shows a visible KL-loss spike, while $\boldsymbol{k_2}$ **as loss** does not. Figure 3 shows this behavior directly: the smoothed KL loss for $\boldsymbol{k_3}$ **as loss** reaches about 0.26, compared with about 0.044 for $\boldsymbol{k_2}$ **as loss**.

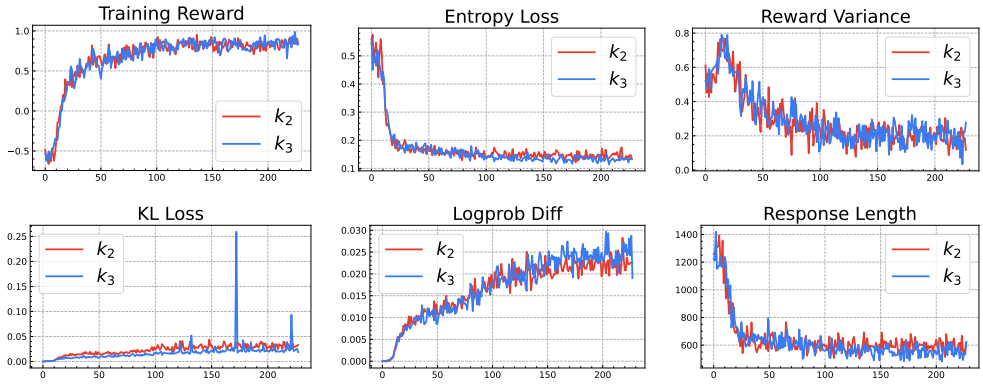

Figure 3: 7B weak-KL training curves at $\beta = 0.001$. The **KL Loss** panel shows a visible k_3 **as loss** spike that is absent for k_2 **as loss**, consistent with the k_3 surrogate's tail sensitivity when drift is larger. The same pattern appears at 1.5B in Figure 6 and at 32B in Figure 12.

Downstream Performance. Table 3 reports benchmark accuracy after training. Methods without effective KL constraints (no KL, k_1 **as loss**) achieve the highest scores, consistent with less-constrained drift; effective KL heads trade some accuracy for tighter reference coupling.

Table 3: Downstream benchmark accuracy (%, $\beta = 0.5$). AIME reports AIME 24 / AIME 25; averages are computed from underlying unrounded scores. See Appendix O for averaging and GPQA* details.

Model / Method	Math Reasoning						General Reasoning			
	AIME 24/25	AMC	MATH-500	Minerva	Olympiad	Avg.	ARC-c	GPQA*	MMLU-Pro	Avg.
Qwen2.5-Math-7B	11.5/4.9	31.3	43.6	7.4	15.6	19.0	18.2	11.1	16.9	15.4
RL w/o KL	20.5/14.4	55.6	78.6	36.8	42.4	41.4	81.7	33.8	46.9	54.1
RL w/ k_1 as loss	19.1/11.6	56.0	80.6	40.8	43.0	41.8	79.7	29.8	45.1	51.5
RL w/ k_2 as loss	15.4/7.5	48.5	64.2	16.9	24.9	29.6	31.3	15.2	27.1	24.5
RL w/ k_3 as loss	19.0/7.3	48.9	65.4	18.8	29.0	31.4	29.6	19.2	27.7	25.5
Qwen2.5-Math-1.5B	7.2/3.6	26.4	28.0	9.6	21.2	16.0	3.5	4.0	2.5	3.3
RL w/o KL	12.5/4.8	43.7	66.8	28.3	31.9	31.3	43.7	19.2	23.1	28.7
RL w/ k_1 as loss	13.8/4.7	41.5	68.0	25.7	31.9	30.9	36.6	18.2	21.0	25.3
RL w/ k_2 as loss	7.0/5.5	35.2	52.8	14.7	29.0	24.0	7.8	7.6	4.9	6.8
RL w/ k_3 as loss	7.7/3.8	34.9	54.2	15.8	28.0	24.1	11.3	8.1	5.5	8.3

Summary and DeepSeek-V3.2. Overall, k_1 **as loss** provides no regularization signal, while k_2 **as loss** implements the RKL-correct decoupled coefficient. k_3 **as loss** is a first-order surrogate whose asymmetric tails can both weaken the restoring force in one direction and create KL-loss spikes in the other. Appendix M further shows that the IS-weighted, gradient-carrying DeepSeek-V3.2 (Liu et al., 2025a) variant cancels the vanilla k_3 **as loss** bias and recovers the RKL-correct k_1 coefficient, at the cost of an unclipped-ratio stability trade-off.

7 Conclusion

This paper develops a gradient-centric framework for KL regularization in RLHF, expressing each implementation as a scalar coefficient on the policy score function. The analysis gives three main findings: (i) k_1 **as loss** has zero expected gradient despite being an unbiased value estimator; (ii) k_1 **in reward** and k_2 **as loss** are gradient-equivalent, theoretically sound implementations of Reverse KL; and (iii) k_3 **as loss** is a biased first-order surrogate with mismatched tail behavior. We also identify missing importance-sampling corrections in off-policy "as loss" implementations. Experiments across model scales and evaluation settings are consistent with these predictions, with k_2 **as loss** showing greater stability than k_3 **as loss** under weak regularization. We recommend k_1 **in reward** or k_2 **as loss** for principled RLHF regularization.

AI USE STATEMENT

We used LLMs for language polishing, editing, and LaTeX formatting and template migration. All retrieval of related work, algorithmic design, and theoretical derivations are carried out by the authors.

ETHICS STATEMENT

This work focuses on the theoretical and empirical analysis of KL regularization in RLHF, using existing datasets and publicly available language models. We do not identify significant ethical concerns specific to the methods and experiments presented in this paper.

REPRODUCIBILITY STATEMENT

The training and evaluation code is included in the supplementary material. Section 6 describes the training data, filtering procedure, model checkpoints, reward construction, and key training settings. Appendices A and J explain how rewards and KL regularization are implemented and how numerical instability in group normalization is handled. The assumptions and detailed proofs underlying our KL-gradient analysis are provided in Appendices C and E, with off-policy corrections derived in Section G.

REFERENCES

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

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:2501.12948*, 2025.

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:2103.03874*, 2021.

Jian Hu, Xibin Wu, Zilin Zhu, Weixun Wang, Dehao Zhang, Yu Cao, et al. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Schulman John. Approximating kl divergence, 2020. URL `http://joschu.net/blog/kl-approx.html`.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, et al. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025.

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

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025a.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *arXiv preprint arXiv:2505.24864*, 2025b.

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

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*, 35:27730–27744, 2022.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Vedant Shah, Johan Obando-Ceron, Vineet Jain, Brian Bartoldson, Bhavya Kailkhura, Sarthak Mittal, Glen Berseth, Pablo Samuel Castro, Yoshua Bengio, Nikolay Malkin, et al. A comedy of estimators: On kl regularization in rl training of llms. *arXiv preprint arXiv:2512.21852*, 2025.

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

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. *arXiv preprint arXiv:2409.19256*, 2024.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

Yunhao Tang and Rémi Munos. On a few pitfalls in kl divergence gradient estimation for rl. *arXiv preprint arXiv:2506.09477*, 2025.

Weixun Wang, Shaopan Xiong, Gengru Chen, Wei Gao, Sheng Guo, Yancheng He, Ju Huang, Jiaheng Liu, Zhendong Li, Xiaoyang Li, et al. Reinforcement learning optimization for large-scale learning: An efficient and user-friendly scaling library. *arXiv preprint arXiv:2506.06122*, 2025.

Xihuai Leo Wang. KL estimator selection in RL: From value unbiasedness to gradient correctness. `https://xihuai18.github.io/reinforcement-learning/2025/12/01/kl-estimators-zh.html`, 2025. Chinese blog post.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

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

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

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An llm post-training framework for rl scaling. `https://github.com/THUDM/slime`, 2025. GitHub repository. Corresponding author: Xin Lv.

APPENDIX CONTENTS

A Detailed Implementation of RLHF Methods **14**

B Policy Gradient Derivation for Reward Maximization **15**

C Formal Proof of the Principled KL Regularization **16**

C.1 Assumptions and Notation 16

C.2 Fundamental Identities 17

C.3 Derivation of the True RKL Gradient 17

C.4 Principled RKL Surrogates: k_1 **in reward** $\Leftrightarrow$ k_2 **as loss** 18

C.5 Conclusion and Implementation Guidance 19

D Surrogate Objective: Full-Vocabulary vs. Monte Carlo **19**

E Formal Analysis of the k_3 as loss Gradient Surrogate **21**

F Derivation of an Alternative Regularizer: The MiniMax-01 Loss **23**

G Off-Policy Correction for KL Regularization **24**

G.1 From On-Policy to Off-Policy Gradients 24

G.2 Correcting k_n as loss by Converting to an In-Reward Head 25

G.3 Two Principled Off-Policy Integration Strategies 25

H Visualization of KL Regularization Gradient Coefficients **26**

I On the Statistical Instability of the k_3 Value Estimator **28**

I.1 Precondition for Unbiasedness 28

I.2 Infinite Variance and the Chi-squared Divergence 28

I.3 The Gaussian Case and an Empirical Demonstration 28

J Group Normalization Stability Issues **30**

K Practical Guidance **31**

L Forward KL vs. Reverse KL: A Gradient-Centric Reinterpretation **31**

M Analysis of the DeepSeek-V3.2 Importance-Weighted KL Loss **33**

M.1 Mathematical Derivation and Trade-off Analysis 33

N Additional 1.5B-Scale Experiments **35**

O Downstream Benchmark Performance **35**

P Additional Validation on Standard Qwen2.5 Base Checkpoints (7B / 32B) **37**

P.1 Qwen2.5-7B 37
P.2 Qwen2.5-32B 37

Q Note on Closely Related Discussions **41**

R Limitations **41**

S Broader Impact Statement **42**

A Detailed Implementation of RLHF Methods

The main text introduces a unified framework that analyzes KL implementations through their induced gradient coefficients. Before diving into the theoretical analysis, it is helpful to establish the practical context: how are RLHF algorithms actually implemented, and where do the different KL formulations fit in?

This section provides that context. We describe the standard minibatch structure, the reward shaping techniques used in practice, and the two canonical ways to integrate KL regularization—the "combined form" (k_n in reward) and the "decoupled form" (k_n as loss). These details will be referenced throughout the theoretical sections that follow.

Data and verification tools. Our main experiments use OpenR1-Math-220k and verify mathematical answers with Math-Verify.

Minibatch structure. In a typical training step, we draw N prompts $\{x^{(i)}\}_{i=1}^{N}$ from the dataset $\mathcal{D}$. For each prompt $x^{(i)}$, we sample G responses $y^{(i,1)}, \ldots, y^{(i,G)} \sim \pi_\theta(\cdot|x^{(i)})$ from the detached policy snapshot. The G responses for each prompt form a *group*; the full minibatch contains $N \times G$ prompt–response pairs. We use $\pi_{\boldsymbol{\theta}}$ for the trainable policy (gradients flow through it) and π_θ for its detached snapshot.

Baseline subtraction and normalization. Having established the minibatch structure, we now describe how the raw reward signal is processed before being used in the policy gradient. A key technique is baseline subtraction: subtracting an action-independent baseline does not change the expected policy gradient (see Equation (7)) and typically reduces variance (Williams, 1992). We consider the following operators applied to a scalar signal r:

$$f_{\text{group-bl}}(r) = r - \text{mean}_{\text{group}}(r),$$
$$f_{\text{batch-bl}}(r) = r - \text{mean}_{\text{batch}}(r).$$

In practice, normalization is also used. Normalization is *not* unbiased but can improve numerical stability by controlling the scale of the reward signal:

$$f_{\text{BN}}(r) = \frac{r - \text{mean}_{\text{batch}}(r)}{\text{std}_{\text{batch}}(r)},$$
$$f_{\text{GN}}(r) = \frac{r - \text{mean}_{\text{group}}(r)}{\text{std}_{\text{group}}(r)}.$$

How common algorithms shape the reward signal. With these operators defined, we can now specify how different RLHF algorithms process the raw reward signal. Let $r_{\text{raw}}(x, y)$ denote the raw score from the reward model. Different algorithms post-process this score as follows:

$$\textbf{REINFORCE:} \quad r(x,y) = f_{\text{BN}}\big(r_{\text{raw}}(x,y)\big),$$
$$\textbf{PPO/REINFORCE++:} \quad r(x,y) = f_{\text{BN}}\big(r_{\text{raw}}(x,y)\big),$$
$$\textbf{GRPO:} \quad r(x,y) = f_{\text{GN}}\big(r_{\text{raw}}(x,y)\big),$$
$$\textbf{Dr-GRPO (Liu et al., 2025c):} \quad r(x,y) = f_{\text{group-bl}}\big(r_{\text{raw}}(x,y)\big),$$
$$\textbf{REINFORCE++baseline:} \quad r(x,y) = f_{\text{BN}}\Big(f_{\text{group-bl}}\big(r_{\text{raw}}(x,y)\big)\Big).$$

Integration of the KL regularization loss. The transforms above shape only the reward signal. A separate and critical design choice is how to integrate the KL regularizer. As discussed in Section 4, there are two canonical approaches:

(i) **Combined form (k_n in reward):** Used by PPO-style and KL-regularized REINFORCE++ implementations. A combined reward signal is formed first,

$$A_{\text{combined}}(x,y) \;=\; r_{\text{raw}}(x,y) \;-\; \beta\, k_1\Big(\pi_\theta(y|x),\, \pi_{\text{ref}}(y|x)\Big),$$

and then baseline/normalization is applied to this combined signal A_{combined} before it multiplies the score function.

(ii) **Decoupled form:** GRPO methods use k_n as loss. The KL penalty $k_n\big(\pi_{\boldsymbol{\theta}}(\cdot|x), \pi_{\text{ref}}(\cdot|x)\big)$ is optimized as a separate, unnormalized loss term. It is added to the policy-gradient loss driven by the shaped reward $r(x, y)$.

Complete on-policy objectives. Combining the reward shaping and KL integration described above, we can now write down the complete on-policy objectives for representative algorithms. These expressions make explicit how the two KL styles manifest in practice.

PPO / KL-regularized REINFORCE++ (Monte Carlo minibatch):

$$\mathcal{L}_{\text{combined,MC}}(\boldsymbol{\theta}) = -\frac{1}{NG}\sum_{i=1}^{N}\sum_{j=1}^{G}\Big\{\, A\big(x^{(i)}, y^{(i,j)}\big)\, \log \pi_{\boldsymbol{\theta}}\big(y^{(i,j)} \mid x^{(i)}\big)\,\Big\},$$
$$\text{where } A(x,y) = f_{\text{BN}}\Big(r_{\text{raw}}(x,y) - \beta\, k_1\Big(\pi_\theta(y|x),\, \pi_{\text{ref}}(y|x)\Big)\Big)\,.$$

Here, the k_1 term is evaluated at the current policy snapshot (no gradients flow through it), consistent with the policy-gradient framework where it acts as a coefficient for the score function.

GRPO (Monte Carlo minibatch):

$$\mathcal{L}_{\text{GRPO,MC}}(\boldsymbol{\theta}) = -\frac{1}{NG}\sum_{i=1}^{N}\sum_{j=1}^{G}\Big\{\, r\big(x^{(i)}, y^{(i,j)}\big)\, \log \pi_{\boldsymbol{\theta}}\big(y^{(i,j)} \mid x^{(i)}\big)\,\Big\}$$
$$+\ \frac{\beta}{NG}\sum_{i=1}^{N}\sum_{j=1}^{G} k_3\Big(\pi_{\boldsymbol{\theta}}\big(y^{(i,j)} \mid x^{(i)}\big),\, \pi_{\text{ref}}\big(y^{(i,j)} \mid x^{(i)}\big)\Big).$$

In this decoupled form, the shaped reward $r(x, y)$ drives the policy-gradient term, while the KL penalty is a separate loss where gradients flow directly through $\pi_{\boldsymbol{\theta}}$ inside $k_3(\cdot)$.

Remark. While baseline subtraction is an unbiased variance-reduction technique, normalization is a biased but often crucial heuristic for practical stability. Both are important engineering details, even if omitted from simplified theoretical analyses.

B Policy Gradient Derivation for Reward Maximization

The gradient-centric framework developed in the main text relies on expressing any KL implementation as a coefficient multiplying the policy score function $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)$. Before analyzing KL regularization specifically, it is instructive to review the standard policy gradient derivation for reward maximization, as the same techniques—the log-derivative trick and the distinction between trainable and detached parameters—apply throughout.

This section derives the policy gradient for the reward-maximization objective, clarifying the distinction between the true objective, its gradient, and surrogate loss functions. We adopt the same notation convention as the main text: $\boldsymbol{\theta}$ denotes parameters that carry gradients, while θ denotes detached parameters (e.g., in the sampling distribution).

Objective. The goal is to maximize expected reward:

$$\mathcal{J}_{\text{reward}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_{\boldsymbol{\theta}}(\cdot|x)}\left[r(x,y)\right] = \mathbb{E}_{x\sim\mathcal{D}}\sum_{y}\left[r(x,y)\cdot\pi_{\boldsymbol{\theta}}(y|x)\right].$$

We assume standard regularity conditions that permit the interchange of differentiation and expectation operators.

Policy gradient derivation. We compute the gradient of the objective function $\mathcal{J}_{\text{reward}}(\boldsymbol{\theta})$ using the log-derivative trick. The distinction between $\boldsymbol{\theta}$ (trainable) and θ (detached) is crucial: the former appears wherever gradients must flow, while the latter marks quantities that are held fixed during

differentiation.

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \mathcal{J}_{\text{reward}}(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{x\sim\mathcal{D}} \sum_y [r(x,y) \cdot \pi_{\boldsymbol{\theta}}(y|x)] \\
&= \mathbb{E}_{x\sim\mathcal{D}} \sum_y [r(x,y) \cdot \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(y|x)] \\
&= \mathbb{E}_{x\sim\mathcal{D}} \sum_y \Big[r(x,y) \cdot \pi_\theta(y|x) \cdot \\
&\qquad \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(y|x)}{\pi_\theta(y|x)} \Big] \\
&= \mathbb{E}_{x\sim\mathcal{D}} \sum_y \pi_\theta(y|x) \, [r(x,y) \cdot \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)] \\
&= \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(\cdot|x)} \left[r(x,y) \cdot \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x) \right].
\end{aligned}$$

The key step occurs in the third line, where we multiply and divide by $\pi_\theta(y|x)$. This serves two purposes: (i) it converts the sum over y into an expectation under π_θ, enabling Monte Carlo estimation; and (ii) at the current iterate, where $\pi_{\boldsymbol{\theta}}$ and its detached snapshot π_θ are numerically equal, it gives the log-derivative identity $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}} / \pi_\theta$. In the final line, the gradient is expressed as an expectation over samples from π_θ, where the sampling process itself has no gradient path—only the score function $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}$ carries gradients. This structure, where a scalar coefficient (without gradients) multiplies the score function, is the foundation for the unified framework in Section 4.

C Formal Proof of the Principled KL Regularization

This section provides the complete proof of Theorem 5.1 from the main text—the central theoretical result establishing that $\boldsymbol{k_1}$ **in reward** and $\boldsymbol{k_2}$ **as loss** are gradient-equivalent implementations of the Reverse KL (RKL) regularizer.

The proof proceeds in four stages. First, we state the regularity assumptions required for the analysis. Second, we establish two fundamental identities—the log-derivative identity and the zero-mean score property—that underpin all policy-gradient derivations. Third, we derive the exact gradient of the true RKL objective. Finally, we show that both surrogates reproduce this target gradient.

Throughout, we use $\pi_{\boldsymbol{\theta}}(\cdot|x)$ for the gradient-carrying policy and $\pi_\theta(\cdot|x)$ for its detached snapshot (numerically identical at the current iterate). Sampling measures and scalar coefficients multiplying the score function are always treated as detached.

C.1 Assumptions and Notation

We begin by stating the regularity assumptions that ensure all subsequent derivations are well-defined. Let $\mathcal{D}$ be a data distribution over prompts x, $\pi_{\text{ref}}(\cdot|x)$ a fixed reference policy, and $\pi_{\boldsymbol{\theta}}(\cdot|x)$ a differentiable policy parameterized by $\boldsymbol{\theta}$. All logarithms are natural.

- **(A1)** **(Valid policy)** For each x, the function $y \mapsto \pi_{\boldsymbol{\theta}}(y|x)$ is a valid probability mass/density: $\pi_{\boldsymbol{\theta}}(y|x) > 0$ on its support, it is differentiable in $\boldsymbol{\theta}$, and normalizes to one, i.e., $\sum_y \pi_{\boldsymbol{\theta}}(y|x) = 1$ (or $\int \pi_{\boldsymbol{\theta}}(y|x)\, dy = 1$).
- **(A2)** **(Interchange)** The interchange of expectation/summation and differentiation is valid (standard regularity conditions).
- **(A3)** **(Independence)** The data distribution $\mathcal{D}$ and reference policy π_{ref} do not depend on the trainable parameters $\boldsymbol{\theta}$.
- **(A4)** **(Support)** The KL divergence is well-defined: for all x and all y in the support of $\pi_\theta(\cdot|x)$, we have $\pi_{\text{ref}}(y|x) > 0$.

Notation convention: Throughout this section, π_θ (black θ) denotes the *detached* copy of $\pi_{\boldsymbol{\theta}}$ (red $\boldsymbol{\theta}$), numerically equal at the current iterate but with no gradient path. Unless stated otherwise, expectations over y are taken with respect to the detached sampling distribution $y \sim \pi_\theta(\cdot|x)$.

C.2 Fundamental Identities

Before deriving the RKL gradient, we establish two lemmas and two corollaries that will be used repeatedly. These identities formalize the key properties of the score function that make policy-gradient methods tractable.

Lemma C.1 (Log-derivative identity with detached denominator)**.** *At the current iterate, for any fixed x and any y with $\pi_\theta(y|x) > 0$,*

$$\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x) = \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(y|x)}{\pi_\theta(y|x)}.$$

Proof. By the chain rule, $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}} = (\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}})/\pi_{\boldsymbol{\theta}}$. At the current iterate, replacing the denominator with its detached, numerically identical copy π_θ preserves the numerical value while making the no-gradient path explicit. □ □

Lemma C.2 (Zero-mean score)**.** *For any fixed x,*

$$\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\right] = 0. \tag{7}$$

Proof. Using Lemma C.1,

$$\sum_y \pi_\theta(y|x)\, \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(y|x)}{\pi_\theta(y|x)} = \sum_y \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(y|x) = \nabla_{\boldsymbol{\theta}} \sum_y \pi_{\boldsymbol{\theta}}(y|x) = \nabla_{\boldsymbol{\theta}}(1) = 0.$$

□

Corollary C.0.1 (Score-function reweighting)**.** *For any function $z(y, x)$ that does not depend on $\boldsymbol{\theta}$,*

$$\sum_y \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(y|x)\, z(y, x) = \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[z(y, x)\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\right].$$

Proof. Multiply and divide by $\pi_\theta(y|x)$, then apply Lemma C.1:

$$\sum_y \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(y|x)\, z(y, x) = \sum_y \pi_\theta(y|x)\, \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(y|x)}{\pi_\theta(y|x)}\, z(y, x) = \sum_y \pi_\theta(y|x)\, z(y, x)\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x).$$

The right-hand side is precisely the expectation $\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}[z(y, x)\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)]$. □

Corollary C.0.2 (Baseline invariance)**.** *For any function $b(x)$ that does not depend on $\boldsymbol{\theta}$,*

$$\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[b(x)\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\right] = b(x) \cdot \mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\right] = 0.$$

Thus, adding an action-independent baseline $b(x)$ (which does not carry gradients) to any coefficient does not change the expected gradient.

C.3 Derivation of the True RKL Gradient

With the fundamental identities in hand, we now derive the gradient of the true RKL objective. This gradient serves as the "target" against which we will compare the two surrogate formulations. The RKL divergence objective is:

$$\mathcal{J}_{\mathrm{RKL}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D}}\left[\sum_y \pi_{\boldsymbol{\theta}}(y|x)\, \log \frac{\pi_{\boldsymbol{\theta}}(y|x)}{\pi_{\mathrm{ref}}(y|x)}\right].$$

Step 1 (Differentiate under the expectation). By (A2), the gradient operator is moved inside the expectation and sum.

Step 2 (Apply product rule). For each y, we differentiate the term $\pi_{\boldsymbol{\theta}}(y|x)\log\pi_{\boldsymbol{\theta}}(y|x)$ using the product rule. Let $f=\pi_{\boldsymbol{\theta}}(y|x)$ and $g=\log\pi_{\boldsymbol{\theta}}(y|x)$. Then:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\big[\pi_{\boldsymbol{\theta}}(y|x)\log\pi_{\boldsymbol{\theta}}(y|x)\big] &= \big(\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(y|x)\big)\,\log\pi_{\boldsymbol{\theta}}(y|x)+\pi_{\boldsymbol{\theta}}(y|x)\cdot\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y|x)\\ &= \big(\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(y|x)\big)\,\big(\log\pi_{\boldsymbol{\theta}}(y|x)+1\big).\end{aligned}$$

After differentiation, we evaluate the scalar multiplier at the detached snapshot, replacing $\log\pi_{\boldsymbol{\theta}}$ by $\log\pi_\theta$ in the coefficient. For the reference term, by (A3) π_{ref} does not depend on $\boldsymbol{\theta}$, so:

$$\nabla_{\boldsymbol{\theta}}\big[-\pi_{\boldsymbol{\theta}}(y|x)\log\pi_{\text{ref}}(y|x)\big] = -(\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(y|x))\,\log\pi_{\text{ref}}(y|x).$$

Step 3 (Collect terms). Combining the two terms from Step 2, we obtain:

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\text{RKL}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D}}\left[\sum_y \nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(y|x)\,\left(\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}+1\right)\right].$$

At this stage, the gradient is expressed as a sum over y, weighted by $\nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(y|x)$. To convert this into a form amenable to Monte Carlo estimation, we apply Corollary C.0.1.

Step 4 (Apply score-function reweighting). Setting the scalar coefficient $z(y,x) := \log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}+1$ (which does not carry gradients) and applying Corollary C.0.1:

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\text{RKL}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\,y\sim\pi_\theta(\cdot|x)}\left[\left(\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}+1\right)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y|x)\right]. \tag{8}$$

Step 5 (Simplify using the zero-mean score property). The coefficient in Equation (8) contains a constant $+1$ term. By Lemma C.2, this term contributes zero to the expected gradient:

$$\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[1\cdot\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y|x)\right]=0.$$

Dropping this zero-contribution term yields the final, simplified gradient:

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\text{RKL}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\,y\sim\pi_\theta(\cdot|x)}\left[\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}\,\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y|x)\right]. \tag{9}$$

C.4 Principled RKL Surrogates: k_1 in reward $\Leftrightarrow$ k_2 as loss

Having derived the true RKL gradient in Equation (9), we now show that two structurally different surrogates—**k_1 in reward** and **k_2 as loss**—both reproduce this exact gradient under on-policy sampling. This equivalence is the core theoretical result that validates both as principled implementations of RKL regularization.

Surrogate 1: k_1 in reward. The first surrogate treats the log-ratio as a scalar coefficient multiplying the score function:

$$\mathcal{J}_{k_1\text{ in reward}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\,y\sim\pi_\theta(\cdot|x)}\Big[\underbrace{\left(\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}\right)}_{\text{coefficient }k_1}\log\pi_{\boldsymbol{\theta}}(y|x)\Big].$$

Since the coefficient k_1 does not carry gradients (it is evaluated at the current snapshot π_θ), differentiating yields:

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}_{k_1\text{ in reward}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\,y\sim\pi_\theta(\cdot|x)}\left[\left(\log\frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}\right)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y|x)\right],$$

which is identical to Equation (9). This is the style used in PPO.

Surrogate 2: k_2 as loss. The second surrogate differentiates *through* the log-ratio via a squared penalty:

$$\mathcal{J}_{k_2 \text{ as loss}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\boldsymbol{\theta}}(\cdot|x)}\Big[\frac{1}{2}\big(\log \pi_{\boldsymbol{\theta}}(y|x) - \log \pi_{\text{ref}}(y|x)\big)^2\Big].$$

Applying the chain rule to differentiate the squared term:

$$\nabla_{\boldsymbol{\theta}}\Big[\frac{1}{2}\big(\log \pi_{\boldsymbol{\theta}}(y|x) - \log \pi_{\text{ref}}(y|x)\big)^2\Big] = \big(\log \pi_{\boldsymbol{\theta}}(y|x) - \log \pi_{\text{ref}}(y|x)\big) \cdot \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x).$$

Evaluating the scalar multiplier at the current policy snapshot π_θ and taking the expectation:

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}_{k_2 \text{ as loss}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\boldsymbol{\theta}}(\cdot|x)}\Big[\big(\log \pi_\theta(y|x) - \log \pi_{\text{ref}}(y|x)\big)\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\Big],$$

which is also identical to Equation (9). Thus, despite their different functional forms, both surrogates induce the same expected gradient—the principled RKL gradient.

C.5 Conclusion and Implementation Guidance

We have now established the central result: under Assumptions (A1)–(A4), the true RKL objective and both surrogates—**k_1 in reward** and **k_2 as loss**—share the same expected gradient, given by Equation (9). This equivalence rests on the following key conventions, which practitioners should adhere to for correct implementation.

Sampling Measure: Samples are drawn from the current policy snapshot $y \sim \pi_\theta(\cdot|x)$ (e.g., via an external rollout engine).

Scalar Coefficients: The scale coefficients that multiply the score function do not carry gradients—they are evaluated at the current snapshot π_θ. Applying Corollary C.0.2, any action-independent baseline $b(x)$ can be added to reduce variance.

Gradient Path: Gradients propagate only through terms explicitly parameterized by $\boldsymbol{\theta}$.

Implementation Notes. We now translate these theoretical results into concrete implementation patterns. For a single on-policy sample $y \sim \pi_\theta(\cdot|x)$:

- **k_1 in reward**: The surrogate loss is

  $$\mathcal{L}_{k_1 \text{ in reward}} = \big(\log \pi_\theta(y|x) - \log \pi_{\text{ref}}(y|x)\big) \cdot \log \pi_{\boldsymbol{\theta}}(y|x),$$

  where the coefficient $k_1(y|x) = \log \pi_\theta(y|x) - \log \pi_{\text{ref}}(y|x)$ is evaluated at the current snapshot (stop gradient) and only $\log \pi_{\boldsymbol{\theta}}(y|x)$ carries gradients. Gradient descent on $\mathcal{L}_{k_1 \text{ in reward}}$ performs descent on $\mathcal{J}_{\text{RKL}}$. In typical RLHF, where we maximize $\mathcal{J}_{\text{reward}} - \beta\, \mathcal{J}_{\text{RKL}}$, the combined loss to minimize is $-\big(r(x,y) - \beta\, k_1(y|x)\big) \cdot \log \pi_{\boldsymbol{\theta}}(y|x)$.
- **k_2 as loss**: The surrogate loss is

  $$\mathcal{L}_{k_2 \text{ as loss}} = \tfrac{1}{2}\big(\log \pi_{\boldsymbol{\theta}}(y|x) - \log \pi_{\text{ref}}(y|x)\big)^2,$$

  where the entire log-ratio carries gradients through $\pi_{\boldsymbol{\theta}}$. Gradient descent on $\mathcal{L}_{k_2 \text{ as loss}}$ also performs descent on $\mathcal{J}_{\text{RKL}}$, since differentiating yields the same coefficient $k_1(y|x) = \log \pi_\theta(y|x) - \log \pi_{\text{ref}}(y|x)$ multiplying $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)$.

Remarks. (i) For continuous spaces, replace sums by integrals; the proof is unchanged provided densities are positive on their support. (ii) The equivalence requires on-policy sampling. If samples are drawn from a stale policy π_{old}, exact correction uses importance weights $\rho(x,y) = \pi_\theta(y|x)/\pi_{\text{old}}(y|x)$ inside the expectations.

D Surrogate Objective: Full-Vocabulary vs. Monte Carlo

The preceding sections derive policy gradients as expectations over the full action space. In practice, however, we cannot enumerate all possible responses y for a given prompt x—the vocabulary is astronomically large for language models. This section bridges the gap between theory and practice

by formalizing the connection between population-level objectives and their minibatch Monte Carlo estimators.

We detail two conceptually distinct implementations: a **full-vocabulary loss**, which computes the exact inner expectation over all actions (theoretically exact but computationally infeasible), and a **Monte Carlo (MC) loss**, which replaces this sum with i.i.d. samples (unbiased and practical). Understanding this connection is essential for verifying that practical implementations are gradient-correct.

Conventions and gradient paths. To ensure consistency with the main text, we restate the key conventions governing gradient flow:

1. The trainable policy $\pi_{\boldsymbol{\theta}}$ carries gradients; its numerically identical snapshot at the current iterate (which does not carry gradients) is denoted π_θ.
2. All scalars that multiply the score function do not carry gradients: the reward $r(x, y)$, any KL-derived term $k_n(\cdot)$, and their combination $r(x, y) - \beta\, k_n(\cdot)$.
3. Gradients flow only through $\log \pi_{\boldsymbol{\theta}}(y|x)$; everything inside the coefficient $c(x, y)$ does not carry gradients.
4. We adopt the naming from the main text: "reward coefficient," "k_n coefficient," and "combined form coefficient."

We express the objective using a generic scalar coefficient $c(x, y)$, which can take several forms:

$$c(x,y) \in \left\{ \begin{array}{ll} r(x,y) & \text{reward coefficient} \\ k_n\big(\pi_\theta(y|x),\, \pi_{\text{ref}}(y|x)\big) & k_n \text{ coefficient} \\ r(x,y) - \beta\, k_n\big(\pi_\theta(y|x),\, \pi_{\text{ref}}(y|x)\big) & \text{combined form coefficient} \end{array} \right\}.$$

A typical KL choice is

$$k_1(y|x) = \log \pi_\theta(y|x) - \log \pi_{\text{ref}}(y|x).$$

Policy gradient in expectation form (with baseline). For the population objective $\mathcal{J}_{\text{true}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\boldsymbol{\theta}}(\cdot|x)}[\, c(x,y)\,]$, using an action-independent baseline $b(x)$ and the log-derivative identity (where the denominator is evaluated at the current snapshot),

$$\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x) = \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(y|x)}{\pi_\theta(y|x)},$$

the unbiased policy gradient is

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_{\text{true}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[(c(x,y) - b(x))\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\Big]. \tag{10}$$

This relies on the zero-mean score property under $y \sim \pi_\theta(\cdot|x)$ as proved in Equation (7), ensuring $b(x)$ does not change the expected gradient.

Population surrogate loss. A surrogate loss whose negative gradient recovers Equation (10) is

$$\mathcal{L}_{\text{sur}}(\boldsymbol{\theta}) = -\,\mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[\big(c(x,y) - b(x)\big)\, \log \pi_{\boldsymbol{\theta}}(y|x)\Big]. \tag{11}$$

Two interchangeable minibatch implementations. We now provide two equivalent minibatch estimators of Equation (11). The first computes the exact inner expectation over the discrete action space $\mathcal{V}$ by summing all actions with a sampling weight—this is the "full-vocabulary" version. The second replaces this inner sum with i.i.d. on-policy samples, yielding an unbiased estimate conditional on the minibatch prompts—this is the "Monte Carlo" version used in practice.

$$\mathcal{L}_{\text{sur,Full}}(\boldsymbol{\theta}) = -\,\frac{1}{N}\sum_{i=1}^{N} \sum_{y^{(i)}\in\mathcal{V}} \underbrace{\pi_\theta\big(y^{(i)} \mid x^{(i)}\big)}_{\text{sampling weight}} \Big(c\big(x^{(i)}, y^{(i)}\big) - b\big(x^{(i)}\big)\Big)\, \log \pi_{\boldsymbol{\theta}}\big(y^{(i)} \mid x^{(i)}\big).$$

Here, the sampling weight π_θ does not carry gradients; the gradient path is solely via $\log \pi_{\boldsymbol{\theta}}$.

$$\mathcal{L}_{\text{sur,MC}}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{G}\sum_{j=1}^{G}\Big(c\big(x^{(i)}, y^{(i,j)}\big) - b(x^{(i)})\Big)\,\log \pi_{\boldsymbol{\theta}}\big(y^{(i,j)} \mid x^{(i)}\big), \qquad y^{(i,j)} \sim \pi_\theta(\cdot|x^{(i)}). \tag{12}$$

In Equation (12), $\{y^{(i,j)}\}_{j=1}^{G}$ are i.i.d. samples from the current policy snapshot $\pi_\theta(\cdot|x^{(i)})$; increasing G reduces variance while preserving unbiasedness.

Unbiasedness and practical considerations. For any fixed $x^{(i)}$ and function f,

$$\mathbb{E}_{\{y^{(i,j)}\}_{j=1}^{G}\ \text{i.i.d.}\sim\pi_\theta(\cdot|x^{(i)})}\left[\frac{1}{G}\sum_{j=1}^{G} f\big(y^{(i,j)}\big)\right] = \sum_{y^{(i)}\in\mathcal{V}} \pi_\theta\big(y^{(i)} \mid x^{(i)}\big)\, f\big(y^{(i)}\big),$$

hence $\mathbb{E}\big[\mathcal{L}_{\text{sur,MC}} \mid \{x^{(i)}\}\big] = \mathcal{L}_{\text{sur,Full}}$. In practice, computing $r(x,y)$ or $k_n(\cdot)$ over the full vocabulary is infeasible for LLMs due to GPU memory constraints; MC estimation is therefore standard.

Alternative decoupled formulation: k_n as loss. In addition to incorporating KL via the coefficient $c(x,y)$, one may add a separate penalty-only loss that differentiates directly through the log-ratio. Let $\psi_n : \mathbb{R} \to \mathbb{R}$ be differentiable (e.g., $\psi_1(t) = t$, $\psi_2(t) = \frac{1}{2}t^2$). Define

$$\mathcal{L}_{k_n\text{ as loss,MC}}(\boldsymbol{\theta}) = \frac{1}{NG}\sum_{i=1}^{N}\sum_{j=1}^{G}\psi_n\Big(\underbrace{\log \pi_{\boldsymbol{\theta}}(y^{(i,j)}|x^{(i)})}_{\text{with grad}} - \underbrace{\log \pi_{\text{ref}}(y^{(i,j)}|x^{(i)})}_{\text{fixed}}\Big). \tag{13}$$

Its gradient takes the score-like form

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{k_n\text{ as loss,MC}}(\boldsymbol{\theta}) = \frac{1}{NG}\sum_{i=1}^{N}\sum_{j=1}^{G}\psi_n'(\cdot)\,\nabla_{\boldsymbol{\theta}}\log \pi_{\boldsymbol{\theta}}(y^{(i,j)}|x^{(i)}),$$

where $(\cdot)$ denotes the log-ratio in Equation (13). Here the prime denotes differentiation with respect to the scalar argument:

$$\psi_n'(t) \triangleq \frac{\mathrm{d}}{\mathrm{d}t}\psi_n(t).$$

Under on-policy sampling, evaluating the scalar coefficient $\psi_n'(\cdot)$ at the current iterate yields the gradient-equivalence established in the main text (see Theorem in Section 5.2 and Appendix C). A common choice $\psi_2(t) = \frac{1}{2}t^2$ recovers the squared log-ratio penalty. The total minimization loss is then $\mathcal{L}_{\text{reward,MC}} + \beta \cdot \mathcal{L}_{k_n\text{ as loss,MC}}$, with π_{ref} fixed (no gradients flow through it), matching the decoupled sign convention in Section 4.

E Formal Analysis of the k_3 as loss Gradient Surrogate

The preceding section established that **k_1 in reward** and **k_2 as loss** are gradient-correct implementations of the RKL regularizer. A natural question is whether other formulations—particularly **k_3 as loss**, which is widely used in GRPO—also implement the same gradient. This section provides the formal analysis showing that they do not.

Specifically, we prove that **k_3 as loss** is a first-order, biased surrogate for the principled RKL gradient, and we rigorously characterize its three core deficiencies:

1. **Local bias**: The induced coefficient differs from the true RKL coefficient except at $\pi_\theta = \pi_{\text{ref}}$.
2. **Tail asymmetry**: The surrogate saturates when the policy over-covers the reference and diverges when it under-covers.
3. **Statistical instability**: The variance of the induced coefficient equals the chi-squared divergence, which can be unbounded.

Setup and notation. Throughout this section, we fix a prompt x and consider on-policy samples $y \sim \pi_\theta(\cdot|x)$. To simplify notation, we define the probability ratio:

$$\delta(y) := \frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)}.$$

The ratio δ captures how much the reference distribution favors action y relative to the current policy: $\delta > 1$ means the reference assigns more probability to y than the current policy, while $\delta < 1$ means the policy assigns more probability than the reference. Our analysis compares the coefficient induced by $\boldsymbol{k_3}$ **as loss** against the principled RKL gradient coefficient, $c^\star(y) = -\log \delta(y)$. All scalar coefficients multiplying the score function $\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)$ are treated as detached.

Gradient-Equivalent Coefficient of k_3 as loss. The $\boldsymbol{k_3}$ **as loss** objective is given by:

$$\mathcal{J}_{k_3 \text{ as loss}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[\frac{\pi_{\text{ref}}(y|x)}{\pi_{\boldsymbol{\theta}}(y|x)} - 1 - \log\frac{\pi_{\text{ref}}(y|x)}{\pi_{\boldsymbol{\theta}}(y|x)}\right].$$

Differentiating and evaluating the resulting scalar multiplier at the detached snapshot yields:

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}_{k_3 \text{ as loss}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[\left(1 - \frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)}\right)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\right].$$

This confirms that $\boldsymbol{k_3}$ **as loss** is gradient-equivalent (under on-policy sampling) to an 'in-reward' update with the following scalar coefficient:

$$c_{3'}(y) := 1 - \delta(y).$$

$$\mathcal{J}_{k_{3'} \text{ in reward}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[\left(1 - \frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)}\right)\log \pi_{\boldsymbol{\theta}}(y|x)\right].$$

The remainder of this section formally analyzes the deficiencies of this proxy, $c_{3'}$, when compared to the principled target, $c^\star = -\log \delta$.

Lemma E.1 (First-order agreement and second-order bias)**.** *The proxy $1 - \delta$ is the first-order Taylor approximation of the principled coefficient $-\log \delta$ around $\delta = 1$. The approximation error (bias) is of second order:*

$$\text{Bias}(\delta) = (-\log \delta) - (1 - \delta) = \frac{1}{2}(\delta - 1)^2 - \frac{1}{3}(\delta - 1)^3 + O\big((\delta - 1)^4\big).$$

Proof sketch. The result is obtained by expanding $-\log \delta$ in a Taylor series at $\delta = 1$ and subtracting the term $(1 - \delta)$. □

Lemma E.2 (One-sided domination and asymmetric tails)**.** *For all $\delta > 0$, the proxy is a strict lower bound, $1 - \delta \le -\log \delta$, with equality holding only at $\delta = 1$. Their tail behaviors are pathologically asymmetric:*

- ***Policy-higher tail** ($\delta \to 0^+$): The proxy provides a weak, saturating restoring force ($\lim_{\delta\to 0^+}(1 - \delta) = 1$), whereas the principled coefficient provides an unbounded penalty ($\lim_{\delta\to 0^+}(-\log \delta) = +\infty$).*
- ***Reference-higher tail** ($\delta \to \infty$): The proxy induces an unbounded* linear *penalty ($\lim_{\delta\to\infty}(1 - \delta) = -\infty$), while the principled coefficient grows only logarithmically.*

Proof sketch. The inequality follows from the fundamental property $\log \delta \le \delta - 1$. The limits are elementary. □

Theorem E.1 (Variance equals chi-squared divergence)**.** *Assuming* $\text{supp}(\pi_{\text{ref}}(\cdot|x)) \subseteq \text{supp}(\pi_\theta(\cdot|x))$*, the proxy coefficient $c_{3'}$ has zero mean under the on-policy sampling distribution, and its variance is exactly the chi-squared divergence:*

$$\mathbb{E}_{y\sim\pi_\theta}[1 - \delta(y)] = 0,$$
$$\text{Var}_{y\sim\pi_\theta}[1 - \delta(y)] = \chi^2(\pi_{\text{ref}}(\cdot|x) \,\|\, \pi_\theta(\cdot|x)).$$

If the support condition is violated, the variance is infinite.

Proof sketch. $\mathbb{E}[\delta] = \sum_y \pi_\theta(y|x) \frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)} = 1$, thus $\mathbb{E}[1-\delta] = 0$. The variance identity then follows directly from the definition of $\chi^2(p \parallel q)$. □

Corollary E.1.1 (Implication for stochastic gradient variance)**.** *The variance of the stochastic gradient term induced by* $\boldsymbol{k_3}$ **as loss** *is directly governed by the chi-squared divergence, an often unstable metric:*

$$\mathbb{E}\left[\|(1-\delta(y))\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\|^2\right] = \mathbb{E}\left[(1-\delta(y))^2 \|\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\|^2\right].$$

Summary. These results provide a rigorous, gradient-centric justification for the claims in the main text:

- The $\boldsymbol{k_3}$ **as loss** formulation does *not* implement the true RKL gradient.
- It deploys a first-order proxy ($c_{3'} = 1 - \delta$) that is accurate only when the policy is very close to the reference ($\delta \approx 1$).
- Its pathological tail behavior—saturating at $+1$ in one direction while diverging to $-\infty$ in the other—introduces optimization challenges not present in the principled $\boldsymbol{k_1}$ **in reward** or $\boldsymbol{k_2}$ **as loss** formulations.
- Its variance equals the chi-squared divergence, which can be unbounded under distribution shift.

This analysis underscores the importance of selecting regularization losses based on their gradient properties, not merely their characteristics as value estimators.

F Derivation of an Alternative Regularizer: The MiniMax-01 Loss

The preceding analysis revealed that KL-based coefficients can be unbounded: the principled RKL coefficient $\log(\pi_\theta(y|x)/\pi_{\text{ref}}(y|x))$ grows without bound as the policy diverges from the reference. While this provides strong regularization, it can also induce large gradient updates that destabilize training in some settings.

This section derives the MiniMax-01 loss (Li et al., 2025) as an alternative regularizer with a *bounded* coefficient. We show that it originates from a mean squared error (MSE) objective in probability space and fits naturally within our gradient-centric framework. The key advantage is that the induced coefficient $\pi_\theta - \pi_{\text{ref}}$ is always bounded in $[-1, 1]$, providing a more conservative regularization force.

Objective. We minimize the full-vocabulary MSE between the policy and the reference:

$$\mathcal{J}_{\text{MSE}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D}}\left[\frac{1}{2}\sum_y \left(\pi_{\boldsymbol{\theta}}(y|x) - \pi_{\text{ref}}(y|x)\right)^2\right]. \tag{14}$$

On-policy gradient. Differentiating Equation (14) with respect to $\boldsymbol{\theta}$, evaluating the scalar multiplier at the detached snapshot π_θ (our standard on-policy convention), and converting to the score-function form yields:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\text{MSE}}(\boldsymbol{\theta}) &= \mathbb{E}_{x\sim\mathcal{D}}\sum_y \left[\frac{1}{2}\nabla_{\boldsymbol{\theta}}(\pi_{\boldsymbol{\theta}}(y|x) - \pi_{\text{ref}}(y|x))^2\right] \\
&= \mathbb{E}_{x\sim\mathcal{D}}\sum_y \underbrace{\left(\pi_\theta(y|x) - \pi_{\text{ref}}(y|x)\right)}_{\text{scalar coefficient}} \nabla_{\boldsymbol{\theta}}\pi_{\boldsymbol{\theta}}(y|x) \\
&= \mathbb{E}_{x\sim\mathcal{D}}\sum_y \pi_\theta(y|x)\left(\pi_\theta(y|x) - \pi_{\text{ref}}(y|x)\right)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y|x) \\
&= \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\left[\left(\pi_\theta(y|x) - \pi_{\text{ref}}(y|x)\right)\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(y|x)\right].
\end{aligned} \tag{15}$$

The last line reveals that MSE regularization induces a score-function update whose scalar coefficient is the probability difference $\pi_\theta - \pi_{\text{ref}}$.

MiniMax-01 surrogate loss (Monte Carlo). Using an on-policy sampler that draws G responses $y^{(i,j)} \sim \pi_\theta(\cdot|x^{(i)})$ per prompt, the unbiased minibatch surrogate whose negative gradient recovers Equation (15) is

$$\mathcal{L}_{\mathrm{MSE,MC}}(\boldsymbol{\theta}) = -\frac{1}{NG}\sum_{i=1}^{N}\sum_{j=1}^{G}\Big(\pi_\theta(y^{(i,j)} \mid x^{(i)}) - \pi_{\mathrm{ref}}(y^{(i,j)} \mid x^{(i)})\Big) \cdot \log \pi_{\boldsymbol{\theta}}(y^{(i,j)} \mid x^{(i)}).$$

This head shares the same in-reward score-function structure as our principled KL implementations: the coefficient does not carry gradients, and only $\log \pi_{\boldsymbol{\theta}}$ contributes to the gradient.

Key properties and implications.

1. Bounded gradient coefficient. Since $0 \le \pi_\theta(y|x), \pi_{\mathrm{ref}}(y|x) \le 1$, the coefficient satisfies $-1 \le \pi_\theta(y|x) - \pi_{\mathrm{ref}}(y|x) \le 1$. This boundedness enhances stability against large or pathological updates, in contrast to the unbounded log-ratio used by KL (see Figure 1). This supports our recommendation in Section 5 to consider bounded alternatives when stability is paramount.
2. Symmetry in probability space. The MSE penalty is symmetric with respect to probability differences, providing more conservative corrections when policies diverge, compared to the logarithmic penalty of Reverse KL.
3. Off-policy compatibility. Owing to its in-reward form (where the coefficient does not carry gradients), this head is fully compatible with importance sampling and clipping, following the same correction rules as in Section G.

Remark. Consistent with our KL analysis, Equation (15) is obtained by evaluating scalar multipliers at the detached snapshot π_θ (on-policy). This expresses the regularizer in the same coefficient-times-score-function form, enabling direct comparison of the induced update dynamics.

G Off-Policy Correction for KL Regularization

All derivations so far have assumed *on-policy* sampling: the samples y are drawn from the current policy π_θ at the moment of gradient computation. In practice, however, many RLHF implementations—particularly PPO with multiple gradient steps per rollout—operate in an *off-policy* setting where samples are drawn from a stale behavior policy π_{θ_k}.

Off-policy updates require importance sampling (IS) and clipping for *both* the reward head and the KL head. When KL is implemented in the combined "in reward" form, the IS correction is inherited automatically from the PPO surrogate. However, when KL is implemented "as loss" in a decoupled form, the required IS/clipping on the KL term is easy to overlook, leading to biased gradients.

This section derives the principled correction and provides concrete guidance for both combined and decoupled objective structures.

G.1 From On-Policy to Off-Policy Gradients

In the policy-gradient view, updates are driven by a scalar coefficient $c(x, y)$ (which does not carry gradients) multiplying the score function. The on-policy gradient estimator is:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_c(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\big[c(x,y)\,\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\big],$$

where π_θ is a detached snapshot numerically equal to $\pi_{\boldsymbol{\theta}}$ at the time of gradient evaluation. For samples drawn from a behavior policy $y \sim \pi_{\theta_k}(\cdot|x)$, an unbiased off-policy estimator requires IS, assuming the behavior policy has support over the sampled data ($\pi_{\theta_k}(y|x) > 0$):

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_c(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\theta_k}(\cdot|x)}\Big[\underbrace{\frac{\pi_\theta(y|x)}{\pi_{\theta_k}(y|x)}}_{\text{IS weight (no gradient)}}\, c(x,y)\,\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\Big].$$

In practice, PPO replaces this detached IS weight with the gradient-carrying ratio $\rho_k(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(y|x)}{\pi_{\theta_k}(y|x)}$ (where gradients flow only through the numerator) and employs a clipped surrogate objective to reduce variance. For any scalar coefficient $c(x, y)$ (which does not carry gradients), the clipped objective to be maximized is:

$$\mathcal{J}_{c,\text{clipped}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\theta_k}(\cdot|x)}\left[\min\Big(\rho_k(\boldsymbol{\theta})\, c(x,y),\ \text{clip}\big(\rho_k(\boldsymbol{\theta}),\, 1-\epsilon,\, 1+\epsilon\big)\, c(x,y)\Big)\right]. \quad (16)$$

G.2 Correcting k_n as loss by Converting to an In-Reward Head

A $\boldsymbol{k_n}$ **as loss** head is gradient-correct only on-policy. To adapt it for off-policy use, it must first be converted to its gradient-equivalent in-reward form. This is achieved by defining a detached (stop-gradient) coefficient $k_{n'}(x, y)$ that reproduces the on-policy gradient of the original loss. For a differentiable penalty $k_n\big(\pi_{\boldsymbol{\theta}}(y|x), \pi_{\text{ref}}(y|x)\big)$, this coefficient is its derivative with respect to the policy's log-probability, evaluated at the snapshot whose local gradient the surrogate is meant to approximate:

$$k_{n'}(x,y) := \frac{\partial}{\partial\, \log \pi_{\boldsymbol{\theta}}(y|x)}\, k_n\big(\pi_{\boldsymbol{\theta}}(y|x), \pi_{\text{ref}}(y|x)\big)\Big|_{\pi_{\boldsymbol{\theta}}=\pi_\theta}.$$

This conversion precisely aligns with the theoretical equivalences established in the main text:

- **Principled $\boldsymbol{k_2}$ as loss**: $k_2(y|x) = \frac{1}{2}(\log \pi_{\boldsymbol{\theta}}(y|x) - \log \pi_{\text{ref}}(y|x))^2 \ \Rightarrow\ k_{2'}(y|x) = \log \pi_\theta(y|x) - \log \pi_{\text{ref}}(y|x)$ (the k_1 in reward coefficient).
- **Proxy $\boldsymbol{k_3}$ as loss**: $k_3(y|x) = \frac{\pi_{\text{ref}}(y|x)}{\pi_{\boldsymbol{\theta}}(y|x)} - 1 - \log \frac{\pi_{\text{ref}}(y|x)}{\pi_{\boldsymbol{\theta}}(y|x)} \ \Rightarrow\ k_{3'}(y|x) = 1 - \frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)}$ (the $k_{3'}$ in reward coefficient).

Once expressed as a scalar coefficient $k_{n'}(x, y)$ (which does not carry gradients), the KL head is handled off-policy exactly like any other score-function head via Equation (16). For exact current-gradient estimation, $k_{n'}$ would be recomputed from the current detached policy. In PPO with multiple epochs per rollout batch, however, the standard local-surrogate convention is to compute $k_{n'}$ once using the rollout policy π_{θ_k} and hold it fixed across epochs, just like advantage estimates. The importance sampling ratio $\rho_k(\boldsymbol{\theta}) = \pi_{\boldsymbol{\theta}}/\pi_{\theta_k}$ still updates with the policy across epochs, providing the off-policy correction for this fixed score-function head.

G.3 Two Principled Off-Policy Integration Strategies

With the correctly derived coefficient $\boldsymbol{k_{n'}}$ in hand, there are two principled ways to integrate it into the PPO objective, mirroring the on-policy discussion in Section 4.

1. Combined Form (Single Clipped Head). Merge the reward advantage and the KL coefficient *before* applying the PPO machinery:

$$A_{\text{combined}}(x,y) \ :=\ r(x,y) \ -\ \beta\, k_{n'}(x,y).$$

The clipped surrogate is then applied to this combined head:

$$\mathcal{J}_{\text{RLHF}}(\boldsymbol{\theta}) = \mathbb{E}_{y\sim\pi_{\theta_k}}\left[\min\Big(\rho_k(\boldsymbol{\theta})\, A_{\text{combined}},\ \text{clip}(\rho_k(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\, A_{\text{combined}}\Big)\right].$$

This is a robust and straightforward approach, as IS and clipping are consistently applied to both components. For correct PPO semantics, form A_{combined} before clipping. If additional baseline subtraction or normalization is applied, it should be reported as an explicit heuristic because it can change the effective scale of the KL weight β.

2. Decoupled Form (Two Clipped Heads). Maintain separate reward and KL objectives, each with its own IS correction and clipping scheme:

$$\mathcal{J}_{\text{reward}}(\boldsymbol{\theta}) = \mathbb{E}_{y\sim\pi_{\theta_k}}\left[\min\Big(\rho_k(\boldsymbol{\theta})\, r(x,y),\ \text{clip}\big(\rho_k(\boldsymbol{\theta}),\, 1-\epsilon_1,\, 1+\epsilon_2\big)\, r(x,y)\Big)\right],$$

$$\mathcal{J}_{\text{KL}}(\boldsymbol{\theta}) = \mathbb{E}_{y\sim\pi_{\theta_k}}\left[\max\begin{pmatrix}\rho_k(\boldsymbol{\theta})\, k_{n'}(x,y),\\ \text{clip}\big(\rho_k(\boldsymbol{\theta}),\, 1-\epsilon,\, 1+\epsilon\big)\, k_{n'}(x,y)\end{pmatrix}\right]. \quad (17)$$

The final objective to maximize is:

$$\mathcal{J}_{\text{RLHF}}(\boldsymbol{\theta}) \;:=\; \mathcal{J}_{\text{reward}}(\boldsymbol{\theta}) \;-\; \beta\, \mathcal{J}_{\text{KL}}(\boldsymbol{\theta}). \tag{18}$$

This decoupled design affords greater flexibility, such as using asymmetric clipping for the reward head (e.g., $\epsilon_2 > \epsilon_1$) to accelerate learning, while retaining conservative, symmetric clipping for the KL head to ensure stable regularization. Baselines or normalization should be applied only to A_{reward}, not to $k_{n'}(x, y)$, to avoid implicitly altering the regularization strength β. Because the KL head is subtracted in Equation (18), the conservative clipped KL surrogate in Equation (17) uses $\max$ rather than $\min$; equivalently, applying the generic clipped PPO surrogate to the coefficient $-\beta k_{n'}$ yields $-\beta \max(\rho_k k_{n'}, \text{clip}(\rho_k) k_{n'})$.

Implementation Notes.

- **Token vs. Sequence Level:** Our derivations use sequence-level probabilities. In token-level PPO, it is often more stable to compute the ratio as $\rho_k = \exp\big(\sum_t \log \pi_{\boldsymbol{\theta}}(y_t \mid \cdot) - \sum_t \log \pi_{\theta_k}(y_t \mid \cdot)\big)$ and apply clipping at the sequence level; per-token clipping can be overly conservative.
- **Masking Consistency:** Sum log-probabilities only over action tokens that contribute to the reward/KL (exclude prompt, padding, or masked tokens) to keep ρ_k aligned with the heads being optimized.
- **Numerical Stability and Support:** Ensure $\pi_{\theta_k}(y|x) > 0$ for all sampled (x, y) and consider numerically capping ρ_k to prevent overflows under extreme ratios.
- **Adaptive β:** If targeting a desired KL via an adaptive schedule, update β outside the gradient path (detached) and avoid mixing it with advantage normalization; adaptation is orthogonal to IS/clipping and works for both combined and decoupled forms.

Sign Convention. We present objectives for *maximization*. Implementations that *minimize* a loss should negate these expressions, e.g., by minimizing $-\mathcal{J}_{\text{combined}}$ or $-(\mathcal{J}_{\text{reward}} - \beta\, \mathcal{J}_{\text{KL}})$.

H Visualization of KL Regularization Gradient Coefficients

The theoretical analysis in Section 5 showed that different KL implementations induce different scalar coefficients multiplying the score function. Visualizing these coefficients as a function of the policy's log-probability provides valuable intuition about their behavior.

This section provides the Python code used to generate Figure 1. The plot compares four coefficient functions:

- The principled RKL coefficient $\log(\pi_\theta(y|x)/\pi_{\text{ref}}(y|x))$ (achieved by $\boldsymbol{k_1}$ **in reward** and $\boldsymbol{k_2}$ **as loss**);
- The biased GRPO coefficient $1 - \pi_{\text{ref}}(y|x)/\pi_\theta(y|x)$ (induced by $\boldsymbol{k_3}$ **as loss**);
- The bounded MiniMax-01 coefficient $\pi_\theta(y|x) - \pi_{\text{ref}}(y|x)$;
- The constant coefficient 1 (induced by $\boldsymbol{k_1}$ **as loss**, which provides no regularization).

The visualization highlights the linear behavior of the principled RKL coefficient versus the asymmetric saturation/unbounded growth of the k_3 surrogate in the tails.

```python
import torch
import matplotlib.pyplot as plt

# --- Plotting Style ---
plt.style.use('seaborn-v0_8-whitegrid')
plt.rcParams.update({
    "text.usetex": False,  # Disable LaTeX rendering
    "font.family": "serif",  # Use a generic serif font
    "font.serif": ["Times New Roman"],  # Specify Times New Roman as the serif font
    "font.size": 14,
```

```
    "axes.labelsize": 16,
    "legend.fontsize": 12,
    "xtick.labelsize": 12,
    "ytick.labelsize": 12,
})

# --- Data Generation ---
log_pi_actor = torch.linspace(-5, 0, steps=400)
pi_actor = torch.exp(log_pi_actor)

pi_ref_val = 0.25
log_pi_ref = torch.log(torch.tensor(pi_ref_val))

# --- Coefficients Calculation ---
coeff_k1_loss = torch.ones_like(log_pi_actor)
coeff_k1_reward = log_pi_actor - log_pi_ref
coeff_k3_loss = 1 - pi_ref_val / pi_actor
coeff_minimax = pi_actor - pi_ref_val

# --- Plotting ---
plt.figure(figsize=(10, 6.5))

plt.plot(log_pi_actor, coeff_k1_reward,
         label=r'$\log\pi_{\theta} - \log\pi_{\mathrm{ref}}$ ($k_1$ in reward / $k_2$ as loss) - Principled',
         color='#808000', linewidth=3, zorder=10)

plt.plot(log_pi_actor, coeff_k3_loss,
         label=r'$1 - \pi_{\mathrm{ref}}/\pi_{\theta}$ ($k_{3^{\prime}}$ in reward / $k_3$ as loss) - Biased Approximation',
         color='Firebrick', linestyle='--', linewidth=2)

plt.plot(log_pi_actor, coeff_minimax,
         label=r'$\pi_{\theta} - \pi_{\mathrm{ref}}$ (MiniMax-01)',
         color='RoyalBlue', linestyle='-.', linewidth=2)

plt.plot(log_pi_actor, coeff_k1_loss,
         label=r'$1$ ($k_1$ as loss) - Zero Expected Gradient',
         color='Gray', linestyle=':', linewidth=2)

plt.axvline(x=log_pi_ref.item(), color='black', linestyle='--', linewidth=1,
            label=r'$\log\pi_{\theta} = \log\pi_{\mathrm{ref}}$')

plt.xlabel(r'Actor Log-Probability: $\log \pi_{\theta}(y|x)$')
plt.ylabel(r'Coefficient of Score Function')
plt.title(r'Comparison of KL Regularization Coefficients ($\pi_{\mathrm{ref}}=0.25$)', fontsize=18)
plt.legend(loc='upper left')

plt.ylim(-4, 4)
plt.xlim(-5, 0)

plt.tight_layout()
plt.savefig('comparison_kl_regularization_coefficients.png', dpi=300, bbox_inches='tight')
plt.show()
```

Listing 1: Python code to generate the comparison plot of KL gradient coefficients.

I On the Statistical Instability of the k_3 Value Estimator

The main text argues that KL implementations should be evaluated by their gradient properties, not their value-estimation properties. However, even when viewed purely as a value estimator, k_3 has significant limitations that are often overlooked.

This section analyzes the statistical properties of the k_3 value estimator. We show that its advertised "unbiasedness" relies on regularity conditions—specifically, absolute continuity of the reference with respect to the sampling distribution—that can fail in practice. When these conditions are violated, k_3 can be biased, and its variance can become infinite. These failure modes provide additional justification for preferring the principled $\boldsymbol{k_1}$ **in reward** or $\boldsymbol{k_2}$ **as loss** formulations.

I.1 Precondition for Unbiasedness

An estimator is unbiased if its expectation equals the true value. For k_3:

$$\mathbb{E}_q[k_3] = \mathbb{E}_q[\delta(x) - 1 - \log \delta(x)] = (\mathbb{E}_q[\delta(x)] - 1) + D_{\mathrm{KL}}(q \parallel p)$$

For k_3 to be unbiased, it is necessary that $\mathbb{E}_q[\delta(x)] = 1$. This condition is met if p is absolutely continuous with respect to q ($p \ll q$), which means that the support of p must be contained within the support of q.

The condition of a finite KL divergence ($D_{\mathrm{KL}}(q \parallel p) < \infty$) is **not sufficient** to guarantee unbiasedness. For example, let q be the uniform distribution in $[0, 1]$ and p be the uniform distribution on $[0, 2]$.

- The KL divergence $D_{\mathrm{KL}}(q \parallel p) = \int_0^1 1 \cdot \log(\frac{1}{0.5}) dx = \log 2$, which is finite.
- However, $\mathbb{E}_q[\delta(x)] = \int_0^1 1 \cdot \frac{p(x)}{q(x)}\, dx = \int_0^1 \frac{0.5}{1}\, dx = 0.5$.
- The estimator expectation is therefore $\mathbb{E}_q[k_3] = (0.5 - 1) + \log 2 = \log 2 - 0.5$, which is biased.

I.2 Infinite Variance and the Chi-squared Divergence

The variance of k_3 is dominated by the second moment of the importance ratio, $\mathbb{E}_q[\delta(x)^2]$. This term is directly related to the Chi-squared divergence.

When $p \ll q$, the identity holds: $\chi^2(p \parallel q) = \mathbb{E}_q[(\delta(x) - 1)^2] = \mathbb{E}_q[\delta(x)^2] - 1$. If p is not absolutely continuous with respect to q ($p \not\ll q$), $\chi^2(p \parallel q)$ is defined to be infinite.

Therefore, the variance of k_3 will be infinite if $\mathbb{E}_q[\delta(x)^2]$ is infinite. This occurs if $p \not\ll q$ or if $p \ll q$ but the tails of q are sufficiently lighter than the tails of p. While the divergence of $\mathbb{E}_q[\delta(x)^2]$ is the primary cause of instability, the finiteness of $\mathrm{Var}(k_3)$ also technically requires the finiteness of $\mathbb{E}_q[(\log \delta(x))^2]$.

I.3 The Gaussian Case and an Empirical Demonstration

For two Gaussian distributions, $p \sim \mathcal{N}(\mu_p, \sigma_p^2)$ and $q \sim \mathcal{N}(\mu_q, \sigma_q^2)$, the variance of k_3 is finite if and only if $\sigma_q^2 > \sigma_p^2/2$. This condition illustrates that the sampling distribution q must be sufficiently "wide" relative to the reference distribution p. This condition generalizes for multivariate Gaussians with covariance matrices Σ_p and Σ_q. **The expectation $\mathbb{E}_q[\delta(x)^2]$ is calculated via an integral involving the ratio of two Gaussian probability densities. For this integral to converge (and thus for the variance to be finite), the matrix $2\Sigma_q - \Sigma_p$ must be positive definite.**

This failure mode is empirically illustrated below, where a narrow Gaussian $q(x)$ ($\sigma_q = 0.2$) is used to estimate the KL divergence to a standard Gaussian $p(x)$ ($\sigma_p = 1$). This configuration violates the condition, since $0.2^2 \not> 1^2/2$.

```
import torch
import torch.distributions as dist

```

```python
# p: reference distribution, q: sampling distribution
p = dist.Normal(loc=0, scale=1)
q = dist.Normal(loc=0.1, scale=0.2) # A narrow distribution where Var[k3] is infinite

# Sample from the narrow distribution q
x = q.sample(sample_shape=(10_000,))

# Ground truth KL divergence D_KL(q || p)
true_kl = dist.kl_divergence(q, p)

# Compute the log-ratio log delta(x) = log(p(x)/q(x))
log_delta = p.log_prob(x) - q.log_prob(x)
delta = torch.exp(log_delta)

# Define estimators
k1 = -log_delta
k2 = log_delta.pow(2) / 2
k3 = delta - 1 - log_delta

# --- Code to generate output ---
print(f"True KL Divergence: {true_kl:.4f}\n")
print("Estimator         | Sample Mean    | Sample Std. Dev.")
print("------------------|----------------|------------------")
estimators = {"k1": k1, "k2": k2, "k3": k3}

for name, k in estimators.items():
    mean = k.mean()
    std = k.std()
    print(f"{name:<17} | {mean:>13.4f} | {std:>16.4f}")

# --- Actual Output 1 ---
True KL Divergence: 1.1344

Estimator         | Sample Mean    | Sample Std. Dev.
------------------|----------------|------------------
k1                |         1.1272 |           0.6912
k2                |         0.8742 |           0.6006
k3                |         0.8136 |           8.8244
# --- Actual Output 2 ---
True KL Divergence: 1.1344

Estimator         | Sample Mean    | Sample Std. Dev.
------------------|----------------|------------------
k1                |         1.1336 |           0.6611
k2                |         0.8611 |           0.5210
k3                |         0.6817 |           4.1082
# --- Actual Output 3 ---
True KL Divergence: 1.1344

Estimator         | Sample Mean    | Sample Std. Dev.
------------------|----------------|------------------
k1                |         1.1348 |           0.6709
k2                |         0.8689 |           0.4968
k3                |         0.6595 |           1.4925
# --- Actual Output 4 ---
True KL Divergence: 1.1344

Estimator         | Sample Mean    | Sample Std. Dev.
------------------|----------------|------------------
k1                |         1.1256 |           0.6962
k2                |         0.8758 |           0.6263
```

```
k3                  |         0.9772 |          26.7379
```

Listing 2: Code illustrating the high variance of the k_3 value estimator when the sampling distribution q(x) is too narrow.

The results illustrate the issue: across repeated trials, the sample standard deviation of k_3 can be substantially larger than that of k_1 and k_2, and the sample mean can deviate noticeably from the true KL value. This discrepancy is not systematic estimator bias, but rather sampling error caused by heavy-tailed variance, implying that far more samples may be needed for reliable estimation in such settings.

J Group Normalization Stability Issues

Beyond the gradient-level issues analyzed in the main text, GRPO's per-prompt group normalization can introduce a separate numerical stability problem. When the variance within a group of responses is very small, normalization can dramatically amplify tiny numerical differences, potentially destabilizing training.

This section describes the issue and proposes a simple fix: clipping the group standard deviation to prevent pathological amplification.

The issue. GRPO normalizes rewards within each group: for a prompt with G responses and rewards $\mathbf{r} = \{r_1, \ldots, r_G\}$, the advantage is

$$A_i = \frac{r_i - \mathrm{mean}_{\mathrm{group}}(\mathbf{r})}{\mathrm{std}_{\mathrm{group}}(\mathbf{r})}.$$

Stability issue. When the within-group variance is very small, normalization can substantially amplify tiny numerical differences. For example, using the unbiased sample standard deviation,

$$\mathbf{r} = [0.99999,\ 1.00001,\ 0.99999,\ 1.00001],$$
$$\mathbf{A} \approx [-0.8660,\ 0.8660,\ -0.8660,\ 0.8660].$$

This can destabilize optimization by turning near-constant rewards into large-magnitude updates.

Proposed solution. Clip the standard deviation to prevent pathological amplification:

$$A_i = \frac{r_i - \mathrm{mean}_{\mathrm{group}}(\mathbf{r})}{\mathrm{clip_std}_{\mathrm{group}}(\mathbf{r})},$$
$$\mathrm{clip_std}_{\mathrm{group}}(\mathbf{r}) = \max\Big(\min(\mathrm{std}_{\mathrm{group}}(\mathbf{r}),\ \mathrm{std}_{\max}),\ \mathrm{std}_{\min}\Big).$$

Here, $\mathrm{std}_{\min} > 0$ is a small floor that prevents exploding normalization when variance collapses, and $\mathrm{std}_{\max}$ avoids under-normalization when variance is unusually large. In practice, setting $\mathrm{std}_{\min}$ as a small constant relative to the reward scale (e.g., 10^{-1}) may be effective.

Why this matters beyond binary rewards. Although binary 0/1 rewards in RLVR can sometimes mitigate extreme cases, more general regression reward models—such as those trained with Bradley–Terry (BT) losses—often produce continuous scores that may become highly concentrated (e.g., near 0 or 1) on easy or very hard prompts. In such regimes, within-group standard deviations can be arbitrarily small even when rewards are bounded in $[0, 1]$, and group normalization will over-scale negligible differences unless a variance floor (or clipping) is used. Therefore, std clipping is important not only for numerical stability but also to avoid over-amplifying noise when reward predictions saturate.

Remark. For reward scores bounded in $[0, 1]$, $\mathrm{std}(\mathbf{r}) < 1$ always holds, but it can be orders of magnitude smaller than 1 in practice; the smaller the variance, the stronger the amplification effect from group normalization. Clipping $\mathrm{std}_{\mathrm{group}}(\mathbf{r})$ preserves the intended scale-invariance when variance is moderate, while guarding against instability when variance collapses.

K Practical Guidance

Based on the analysis in Section 5, we offer the following practical guidance and test these predictions empirically in Section 6:

- **Avoid k_1 as loss.** Its expected gradient is zero and independent of the reference—it provides no regularization, only noise.
- **Use k_1 in reward or k_2 as loss for RKL regularization.** These are gradient-equivalent under on-policy sampling and correctly implement the RKL objective.
- **Recognize k_3 as loss as a biased surrogate.** It can impose weaker constraints when drift is large and may induce unbounded updates in the opposite tail.
- **Apply IS/clipping in off-policy settings.** When using k_n as loss with PPO-style updates, explicit importance sampling and clipping are required for the KL head; omitting them introduces systematic bias (see Section G).
- **Consider bounded alternatives if stability is paramount.** The MSE-based penalty (e.g., MiniMax-01 loss) induces a coefficient bounded in $[-1, 1]$ (see Appendix F).

L Forward KL vs. Reverse KL: A Gradient-Centric Reinterpretation

The main text focuses on the Reverse KL (RKL) divergence—the standard regularization objective in RLHF—and shows that **k_3 as loss** is a biased first-order surrogate for the RKL gradient. An alternative perspective, recently noted by Tang & Munos (2025), is that k_3 as loss actually optimizes a Forward KL (FKL) objective, $D_{\mathrm{KL}}(\pi_{\mathrm{ref}} \| \pi_\theta)$.

This section refines this observation using our gradient framework. We show that **k_3 as loss** can be written as a *variance-reduced estimator* of the FKL gradient with an implicit baseline of -1. This explains two empirical observations without changing the main conclusion that it is not a principled RKL surrogate:

- **Low variance near the reference**: When $\pi_\theta \approx \pi_{\mathrm{ref}}$, the coefficient $1 - \delta$ fluctuates near zero, giving low variance.
- **Weak constraint under drift**: In the direction where the policy assigns much more probability than the reference, the induced coefficient saturates instead of growing with the RKL coefficient.

These properties make **k_3 as loss** behave well near the reference but provide weaker reference coupling when the policy drifts—consistent with the training metrics in Section 6.

Derivation of the FKL Gradient. The Forward KL objective is defined as:

$$\mathcal{J}_{\mathrm{FKL}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D}}\Big[D_{\mathrm{KL}}\big(\pi_{\mathrm{ref}}(\cdot|x) \,\|\, \pi_{\boldsymbol{\theta}}(\cdot|x)\big)\Big] = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\mathrm{ref}}(\cdot|x)}\big[\log \pi_{\mathrm{ref}}(y|x) - \log \pi_{\boldsymbol{\theta}}(y|x)\big].$$

Rewriting the gradient expectation using importance sampling over the current policy π_θ (to allow on-policy estimation) yields the standard FKL policy gradient:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_{\mathrm{FKL}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[\underbrace{-\frac{\pi_{\mathrm{ref}}(y|x)}{\pi_\theta(y|x)}}_{\text{Standard FKL coeff.}(-\delta)} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\Big], \qquad \text{where } \delta := \frac{\pi_{\mathrm{ref}}(y|x)}{\pi_\theta(y|x)}. \tag{19}$$

k_3 as Loss: FKL with an Implicit Baseline. Recalling Equation (6), the gradient induced by the k_3 as loss formulation uses the coefficient $1 - \delta$. Comparing this with the standard FKL coefficient in Equation (19) reveals an implicit decomposition:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_{k_3 \text{ as loss}}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[\big(\underbrace{(-\delta)}_{\text{Standard FKL}} - \underbrace{(-1)}_{\text{Implicit Baseline } b}\big) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\Big].$$

Although mathematically equivalent in expectation (since $\mathbb{E}[\nabla \log \pi] = 0$), the implicit baseline $b = -1$ acts as a **control variate**:

- **Variance Reduction at Convergence:** Near the reference policy ($\pi_\theta \approx \pi_{\text{ref}}$), we have $\delta \approx 1$. The standard FKL estimator $-\delta$ fluctuates around -1 (high variance), whereas the k_3 coefficient $(1 - \delta)$ fluctuates around 0. This explains why k_3 **exhibits low variance specifically** in the low-KL regime: it acts as a variance-reduced estimator near the reference policy.

Coefficient Implications for Drift Control. While the implicit baseline stabilizes estimation near the reference, the induced coefficient differs from the RKL coefficient away from the reference:

- **RKL coefficient:** As derived in Section 5.2, RKL uses $-\log \delta$. As $\delta \to 0$ (the policy assigns much more probability than the reference), $-\log \delta \to \infty$, so the restoring coefficient grows with the amount of drift.
- k_3**/FKL-like coefficient:** The k_3 coefficient $1 - \delta$ saturates at 1 as $\delta \to 0$. Consequently, in this drift direction, $\boldsymbol{k_3}$ **as loss** supplies a finite coefficient where the principled RKL coefficient would keep growing, explaining why the surrogate can behave differently from an RKL-correct implementation under drift.

In summary, $\boldsymbol{k_3}$ **as loss** can be seen as a statistically coherent, baseline-corrected estimator of the FKL gradient, with favorable variance properties near the reference policy. However, this is a different objective-level interpretation, not evidence that $\boldsymbol{k_3}$ **as loss** is a better RKL estimator. For RLHF applications where the primary goal is to control drift from the reference model, the principled RKL implementations discussed in Section 5.2 offer a more direct regularization target.

This interpretation should not be read as a recommendation to use $\boldsymbol{k_3}$ **as loss** for standard RLHF regularization. FKL is often described using distribution-fitting terminology, but those descriptions do not translate directly into sampled LLM RLHF behavior. Our point is narrower: if a practitioner chooses to replace the standard reference-anchored RKL objective with an FKL-like objective, then $\boldsymbol{k_3}$ **as loss** should be understood in that different-objective sense, not as a better estimator of RKL. For stability- and safety-oriented RLHF settings where the goal is to control drift from the reference policy, our recommendation remains to use $\boldsymbol{k_1}$ **in reward** or its gradient-equivalent decoupled form, $\boldsymbol{k_2}$ **as loss**.

k_3 **in reward: a mixed (and counterproductive) gradient.** A natural question is whether the nonnegative single-sample term k_3 can be used safely as an *in-reward coefficient* (i.e., a scalar multiplier without gradients). Our framework shows it cannot: unlike k_1, it does not induce the RKL gradient.

Let $\delta = \pi_{\text{ref}}(y|x)/\pi_\theta(y|x)$. By definition, $k_3 = \delta - 1 - \log \delta = k_1 + (\delta - 1)$ with $k_1 = -\log \delta$. Therefore, under on-policy sampling,

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathcal{J}_{k_3 \text{ in reward}}(\boldsymbol{\theta}) &= \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[(k_1 + \delta - 1)\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\Big] \\ &= \nabla_{\boldsymbol{\theta}} \mathcal{J}_{\text{RKL}}(\boldsymbol{\theta}) \;+\; \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[\delta\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\Big],\end{aligned}$$

where the constant term -1 drops by the zero-mean score identity. The remaining extra term rewrites as

$$\mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}\Big[\delta\, \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\Big] = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_{\text{ref}}(\cdot|x)}\Big[\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)\Big] = -\nabla_{\boldsymbol{\theta}} \mathcal{J}_{\text{FKL}}(\boldsymbol{\theta}),$$

where the last equality follows from Equation (19). Hence,

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_{k_3 \text{ in reward}}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathcal{J}_{\text{RKL}}(\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \mathcal{J}_{\text{FKL}}(\boldsymbol{\theta}).$$

Equivalently, using k_3 as a reward penalty corresponds to optimizing a *difference* of KL objectives, RKL $-$ FKL, not their sum. In RLHF, this adds an *anti-FKL* component that can counteract the intended constraint and exacerbate instability. We therefore recommend using k_1 (or its gradient-equivalent surrogates) for reward shaping instead.

M Analysis of the DeepSeek-V3.2 Importance-Weighted KL Loss

The off-policy correction framework in Section G showed that $\boldsymbol{k_n}$ **as loss** implementations require explicit importance sampling (IS) and clipping when samples are drawn from a stale policy. A natural question is whether alternative IS constructions can avoid these complications.

DeepSeek-V3.2 (Liu et al., 2025a) proposes an IS-corrected k_3 estimator for off-policy KL regularization. Interestingly, PPO-style clipping is applied only to the reward head while the KL term is left **unclipped**. This section analyzes this construction through our gradient-centric lens and highlights a practical trade-off: under full differentiation, the estimator recovers the principled k_1 coefficient through cancellation; however, leaving the IS-weighted KL term unclipped can introduce numerical instability when the policy diverges from the behavior policy.

The estimator. DeepSeek defines the following IS-weighted k_3 loss (we use $\mathcal{L}_{\mathrm{KL,DS}}$ to distinguish it from the true KL divergence D_{KL}):

$$\mathcal{L}_{\mathrm{KL,DS}}(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(y|x)}{\pi_{\mathrm{old}}(y|x)} \left(\frac{\pi_{\mathrm{ref}}(y|x)}{\pi_{\boldsymbol{\theta}}(y|x)} - \log \frac{\pi_{\mathrm{ref}}(y|x)}{\pi_{\boldsymbol{\theta}}(y|x)} - 1 \right). \tag{20}$$

Our gradient-centric analysis shows that, under full differentiation, this IS-corrected k_3 construction is mathematically equivalent to an in-reward k_1 coefficient, while leaving the IS-weighted KL term unclipped can be numerically fragile. We highlight two issues:

Mathematical equivalence. As derived below, the gradient of the fully differentiated expectation $\mathbb{E}[\rho(\boldsymbol{\theta}) \cdot k_3(\boldsymbol{\theta})]$ equals the exact RKL penalty gradient:

$$\nabla_{\boldsymbol{\theta}} \mathcal{J}_{\mathrm{RKL}} = \mathbb{E}\left[\underbrace{\rho(\theta) \cdot \left(-\log \frac{\pi_{\mathrm{ref}}(y|x)}{\pi_{\theta}(y|x)} \right)}_{\text{Evaluated Scalar Coefficient}} \cdot \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x) \right].$$

DeepSeek's estimator achieves this through cancellation of terms, effectively reconstructing the k_1 coefficient $\log(\pi_\theta(y|x)/\pi_{\mathrm{ref}}(y|x)) = -\log(\pi_{\mathrm{ref}}(y|x)/\pi_\theta(y|x))$. Thus, under full differentiation, it is mathematically identical to the standard $\boldsymbol{k_1}$ **in reward** formulation (without clipping), while requiring additional computation.

Practical trade-off with unclipped IS weights. Leaving the IS-weighted KL term **unclipped** creates a practical trade-off:

- **Full-gradient case:** If the gradient path through $\rho(\boldsymbol{\theta})$ is preserved (so that Part A + Part B terms are both present), the resulting coefficient scales with the potentially large importance weight $\rho(\theta)$. Without clipping, this can yield very large gradients when the policy diverges from π_{old}.
- **Detached-weight case:** If ρ is detached for stability, the estimator drops the sampling-distribution gradient term and reduces to the biased GRPO direction $(1 - \pi_{\mathrm{ref}}(y|x)/\pi_\theta(y|x))$, losing the equivalence to the RKL gradient.

M.1 Mathematical Derivation and Trade-off Analysis

We analyze the gradient of the KL estimator in Equation (20). Let $\pi_{\boldsymbol{\theta}}$ denote the trainable policy carrying gradients, and π_{old} denote the behavior policy (detached).

Full Gradient Derivation (Equivalence to $\boldsymbol{k_1}$). The objective function corresponding to Equation (20) is:

$$\mathcal{L}_{\mathrm{KL,DS}}(\boldsymbol{\theta}) = \mathbb{E}_{y \sim \pi_{\mathrm{old}}(\cdot|x)} \left[\underbrace{\frac{\pi_{\boldsymbol{\theta}}(y|x)}{\pi_{\mathrm{old}}(y|x)}}_{\rho(\boldsymbol{\theta})} \cdot \underbrace{\left(\frac{\pi_{\mathrm{ref}}(y|x)}{\pi_{\boldsymbol{\theta}}(y|x)} - \log \frac{\pi_{\mathrm{ref}}(y|x)}{\pi_{\boldsymbol{\theta}}(y|x)} - 1 \right)}_{k_3(\boldsymbol{\theta})} \right].$$

Applying the product rule $\nabla_{\boldsymbol{\theta}}(f \cdot g) = g\nabla_{\boldsymbol{\theta}} f + f\nabla_{\boldsymbol{\theta}} g$:

$$\nabla_{\boldsymbol{\theta}}\big(\rho(\boldsymbol{\theta}) \cdot k_3(\boldsymbol{\theta})\big) = \underbrace{(\nabla_{\boldsymbol{\theta}}\rho(\boldsymbol{\theta})) \cdot k_3(\theta)}_{\text{Part A}} + \underbrace{\rho(\theta) \cdot (\nabla_{\boldsymbol{\theta}} k_3(\boldsymbol{\theta}))}_{\text{Part B}}.$$

Note that in the product rule expansion, the term not being differentiated is treated as an evaluated value (hence black θ).

Part A (from $\nabla_{\boldsymbol{\theta}}\rho$): Using the log-derivative trick $\nabla_{\boldsymbol{\theta}}\rho(\boldsymbol{\theta}) = \rho(\theta) \cdot \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x)$:

$$\text{Part A} = \rho(\theta) \cdot \underbrace{\left(\frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)} - \log \frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)} - 1\right)}_{k_3 \text{ scalar value}} \cdot \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x).$$

Part B (from $\nabla_{\boldsymbol{\theta}} k_3$): Differentiating k_3 with respect to $\boldsymbol{\theta}$ yields the biased GRPO coefficient:

$$\nabla_{\boldsymbol{\theta}} k_3(\boldsymbol{\theta}) = \left(1 - \frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)}\right) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x),$$

and therefore:

$$\text{Part B} = \rho(\theta) \cdot \left(1 - \frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)}\right) \cdot \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x).$$

Total Gradient (Cancellation): Summing Part A and Part B, the bias terms in the coefficients cancel perfectly:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{KL,DS}} &= \rho(\theta) \cdot \left[\left(\frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)} - \log \frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)} - 1\right) + \left(1 - \frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)}\right)\right] \cdot \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x) \\ &= \rho(\theta) \cdot \underbrace{\left(-\log \frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)}\right)}_{\text{evaluated } k_1 \text{ coefficient}} \cdot \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x).\end{aligned}$$

Conclusion. Under full differentiation, this construction recovers the same score-function coefficient as $\boldsymbol{k_1}$ **in reward** (up to the IS weight). In this sense, it does not provide a different regularization direction; it realizes the k_1 coefficient through cancellation within a more complex expression.

A practical trade-off with unclipped IS weights. DeepSeek-V3.2 separates this term and leaves it **unclipped**. This leads to the following trade-off:

Case 1 (full gradient path). If $\rho(\boldsymbol{\theta})$ is kept active (i.e., both Part A and Part B are included), the gradient magnitude scales with $\rho(\theta)$. When the policy diverges from π_{old} (e.g., $\pi_\theta \gg \pi_{\text{old}}$), $\rho(\theta)$ can become large; without clipping, this can produce very large gradients and numerical instability.

Case 2 (detached weight). If ρ is detached for stability, then $\nabla_{\boldsymbol{\theta}}\rho \equiv 0$ and Part A vanishes. The estimator reduces to Part B:

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{detached}} = \rho(\theta) \cdot \left(1 - \frac{\pi_{\text{ref}}(y|x)}{\pi_\theta(y|x)}\right) \cdot \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(y|x).$$

This recovers the **biased GRPO coefficient** $1 - \pi_{\text{ref}}(y|x)/\pi_\theta(y|x)$, rather than the principled k_1 coefficient.

Summary. This analysis highlights a practical tension when using unclipped, IS-weighted decoupled KL terms: preserving the full gradient path can lead to large updates when ρ is large, while detaching ρ changes the regularization direction. The principled approaches in Section 5.2—$\boldsymbol{k_1}$ **in reward** with standard PPO correction, or the gradient-equivalent $\boldsymbol{k_2}$ **as loss**—avoid this particular failure mode.

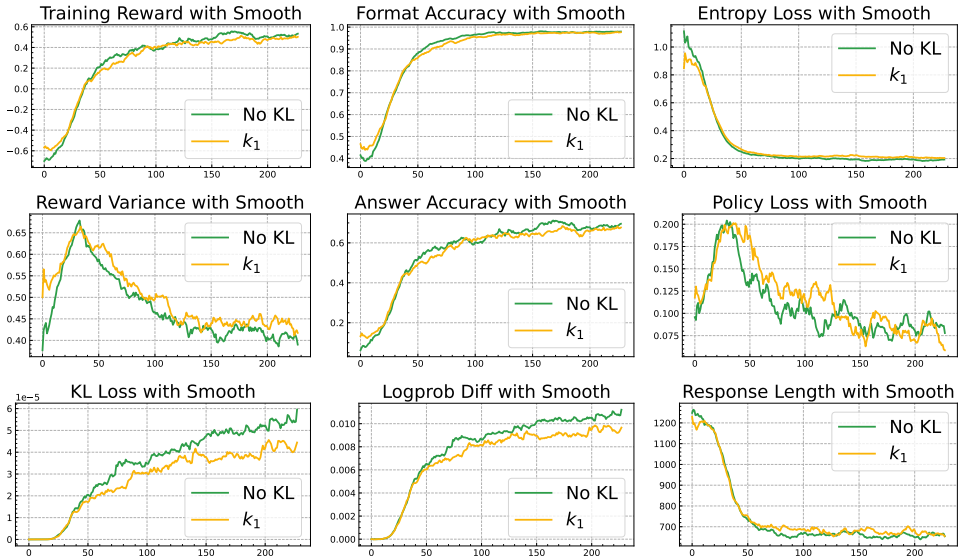

Figure 4: 1.5B-scale comparison of k_1 **as loss** versus no KL regularization. The gap is smaller than at 7B, but k_1 **as loss** still does not provide reliable regularization and can add stochasticity, consistent with its zero-mean, reference-independent expected gradient (Section 5).

N Additional 1.5B-Scale Experiments

The main text presents experimental results at 7B scale. To assess the generality of our findings, this section provides complementary experiments at 1.5B scale using the Qwen2.5-Math-1.5B model.

The 1.5B experiments test the same hypotheses as the main text: (i) k_1 **as loss** should provide no meaningful regularization, and (ii) k_2 **as loss** and k_3 **as loss** should differ according to their induced coefficients. Consistent with our theoretical analysis, we find that these patterns hold at 1.5B scale, though the differences between methods are somewhat less pronounced than at 7B. The two main comparisons are shown in Figures 4 and 5.

Compared to the 7B-scale behavior in the main text (Figure 2), the 1.5B runs separate less. This matches our analysis: k_1 **as loss** mainly behaves as a zero-mean perturbation, so its qualitative effect is smaller at 1.5B and more pronounced at 7B in our runs.

Weaker KL weight ($\beta = 0.001$). We also conduct experiments at 1.5B scale with a weaker practical KL weight ($\beta = 0.001$) to test the sensitivity of different surrogates under larger drift. As shown in Figure 6, the **KL Loss** panel reveals a clear loss spike for k_3 **as loss**, while k_2 **as loss** does not show the same spike. This pattern mirrors the 7B results in Figure 3, suggesting that the tail sensitivity of the k_3 surrogate under distribution shift is consistent across model scales.

O Downstream Benchmark Performance

The main text reports the full Qwen2.5-Math downstream evaluation in Table 3. This section defines the table metrics, highlights the main patterns, and connects them to the additional standard-base validation in Appendix P.

Evaluation and averaging details. Table 3 summarizes downstream performance for 7B and 1.5B models across six math benchmark scores (AIME 24, AIME 25, AMC, MATH-500, Minerva, and Olympiad) and three general reasoning benchmarks (ARC-c, GPQA, and MMLU-Pro). The AIME column reports AIME 24 / AIME 25 separately. The math "Avg." column averages AIME 24, AIME 25, AMC, MATH-500, Minerva, and Olympiad with equal weight; the general "Avg." column averages ARC-c, GPQA*, and MMLU-Pro with equal weight. All averages are computed from the underlying unrounded scores before display rounding. GPQA* denotes the fixed GPQA

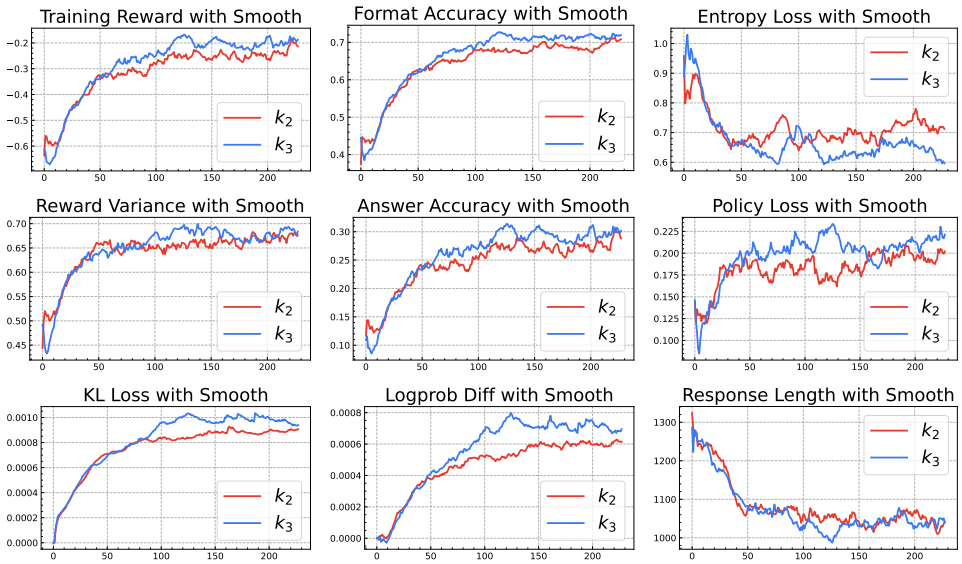

Figure 5: 1.5B-scale comparison of the principled k_2 **as loss** against its first-order surrogate k_3 **as loss**. The surrogate's asymmetric tails can weaken the restoring force in one direction and make the opposite tail spike-prone.

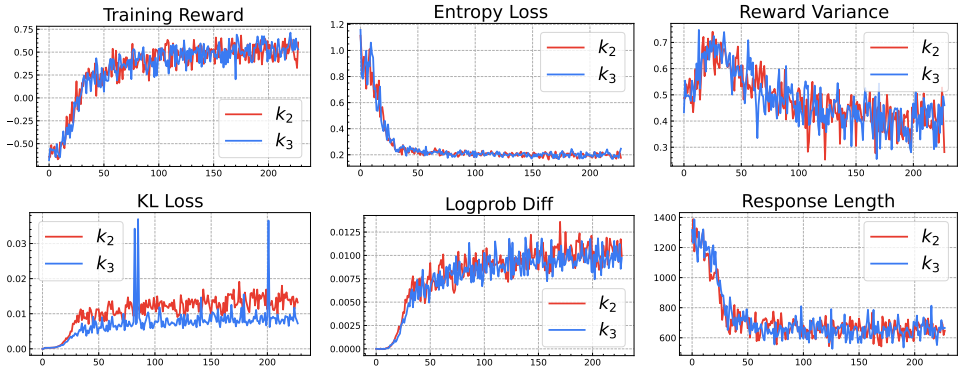

Figure 6: 1.5B-scale training metrics for k_2 **as loss** and k_3 **as loss** at a weaker KL weight ($\beta = 0.001$). The **KL Loss** panel (bottom-left) shows a visible spike for k_3 **as loss** that is absent for k_2 **as loss**, consistent with the 7B-scale results in Figure 3. This supports the interpretation that the k_3 surrogate's tail sensitivity under distribution shift persists across model scales.

evaluation setting used in our benchmark suite; all methods and model scales use the same setting, so the GPQA* column is intended for within-table comparison.

Several patterns emerge that are consistent with our theoretical analysis:

k_1 as loss provides no regularization benefit. Performance is close to the no-KL baseline, consistent with our analysis: its expected gradient vanishes and is independent of the reference model.

k_2 as loss and k_3 as loss trade off differently. The principled **k_2 as loss** implements the RKL-correct decoupled coefficient, while **k_3 as loss** is a first-order surrogate with asymmetric tails: its coefficient saturates in the high-policy-mass direction, weakening the restoring force, and its opposite tail can produce large KL-loss spikes. In this setting, stronger reference-directed regularization correlates with lower downstream scores on both math benchmarks (AIME (Li et al., 2024), AMC, MATH-500 (Hendrycks et al., 2021)) and general reasoning benchmarks (ARC-c (Clark et al., 2018), GPQA* (Rein et al., 2024), MMLU-Pro (Wang et al., 2024)).

Summary. These results align with our theoretical analysis: k_1 **as loss** does not constrain; k_2 **as loss** implements the principled RKL coefficient; k_3 **as loss** is a first-order surrogate whose asymmetric tails can both under-regularize policy-mass increases and create KL-loss spikes.

P Additional Validation on Standard Qwen2.5 Base Checkpoints (7B / 32B)

The main experiments use Qwen2.5-Math checkpoints, which have been separately optimized for mathematical reasoning. To further test whether our gradient-based predictions are tied to this specialized initialization, we additionally evaluate the standard Qwen2.5-7B and Qwen2.5-32B base checkpoints. These checkpoints have not undergone the same math-specific post-training, but they still retain strong mathematical reasoning ability; we therefore use them as a complementary testbed, not as weak baselines. The goal is to examine whether the same KL implementation effects persist under standard base-checkpoint initialization and at a larger parameter scale. All other training settings—dataset, reward, learning rate, KL-integration form, and the total number of prompts consumed—are matched to the 7B setup described in Section 6 unless noted otherwise.

Each figure reports the same nine training metrics in a 3×3 grid, ordered row-wise as follows: training reward, format reward, entropy loss; group-level reward standard deviation, score, policy loss; KL loss, the empirical k_1 gap, and response length.

As in the Qwen2.5-Math experiments, the qualitative patterns are unchanged: k_1 **as loss** does not effectively constrain drift across $\beta \in \{0, 0.001, 0.5\}$; k_2 **as loss** implements the RKL-correct decoupled coefficient; and the k_3 surrogate shows KL-loss spikes under weak KL weights that k_2 **as loss** avoids. This spike behavior is the empirical signature of the asymmetric tail analysis: one tail supplies only a saturated restoring coefficient, while the other can generate very large loss values.

P.1 Qwen2.5-7B

Two of the three figures in this subsection use 228-step runs with rollout batch size 32. The k_2 **as loss** versus k_3 **as loss** comparison at $\beta = 0.001$ uses 114-step runs with rollout batch size 64. This pair consumes the same total number of prompts (7,296) as the 228-step runs, so the weak-β comparison is data-matched; we use it only for the targeted k_2 **as loss** versus k_3 **as loss** comparison.

Experiment 1: k_1 as loss versus no KL across β. Figure 7 compares no-KL training against k_1 **as loss** at $\beta = 0.5$ and $\beta = 0.001$ on Qwen2.5-7B. The three curves stay close on reward, score, and format metrics; changing β does not create a reference-directed regularization effect because the induced coefficient of k_1 **as loss** is independent of π_{ref}. The empirical drift metrics (k_1 gap and response length) are therefore not controlled in the way an RKL penalty should be. This reproduces the main-text trend (Figure 2) under the standard Qwen2.5 initialization.

Experiment 2: k_2 as loss versus k_3 as loss at $\beta = 0.5$. Figure 8 shows the $\beta = 0.5$ comparison on Qwen2.5-7B (228-step runs). The principled k_2 **as loss** follows the RKL-correct decoupled coefficient, while the k_3 **as loss** surrogate inherits the same asymmetric tail structure analyzed above. The lower downstream accuracy of k_2 **as loss** in Table 4 is consistent with a regularization trade-off, rather than a failure of the gradient analysis.

Weaker KL weight ($\beta = 0.001$). Figure 9 shows the $\beta = 0.001$ comparison (114-step rollout-batch-64 runs). Because the KL weight is much weaker, most reward-side metrics are close. The key difference appears in the **KL Loss** panel: k_3 **as loss** exhibits the spike observed in Figure 3, while k_2 **as loss** does not show the same spike on the data-matched schedule.

P.2 Qwen2.5-32B

We further repeat the two experiments on the larger Qwen2.5-32B base model. All runs use rollout batch size 64; complete runs therefore reach 114 steps. The $\beta = 0.5$ k_1 **as loss** run terminated after 67 steps due to a CUDA out-of-memory error; all other curves are complete (114 steps). We therefore

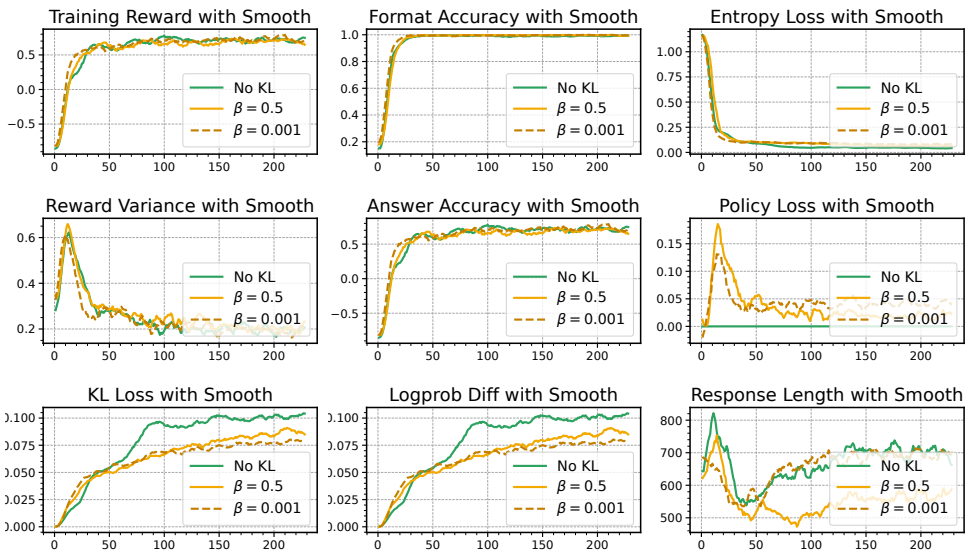

Figure 7: Qwen2.5-7B: no KL vs. k_1 **as loss** with $\beta \in \{0.5, 0.001\}$ over 228 steps. Panels are ordered row-wise as reward, format reward, entropy loss; group reward standard deviation, score, policy loss; KL loss, empirical k_1 gap, and response length. Varying β does not turn k_1 **as loss** into an effective reference-coupling term, consistent with its zero-mean, reference-independent gradient.

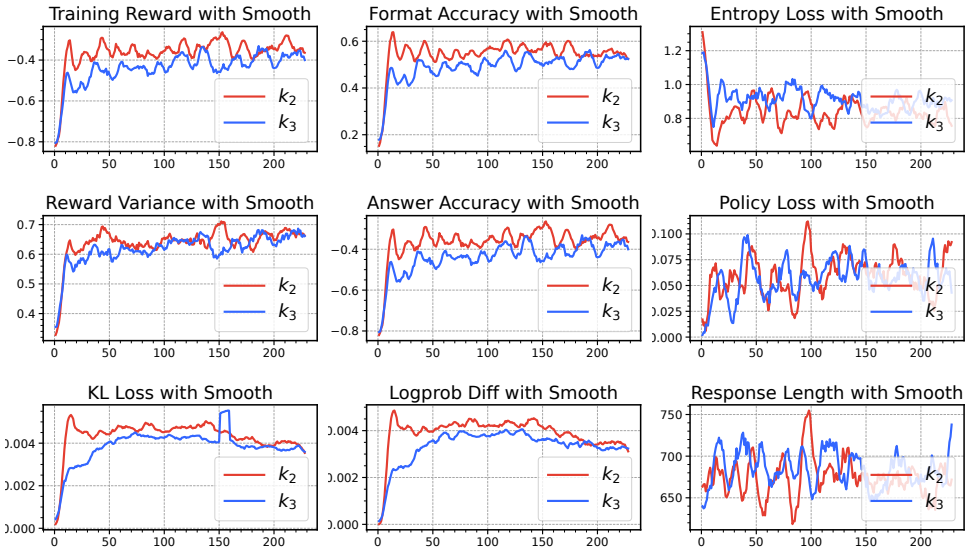

Figure 8: Qwen2.5-7B: k_2 **as loss** vs. k_3 **as loss** at $\beta = 0.5$ over 228 steps. The comparison contrasts the RKL-correct decoupled coefficient with the first-order k_3 **as loss** surrogate, whose asymmetric tails weaken one drift direction and make the opposite tail spike-prone.

use the partial k_1 **as loss** curve only to inspect training dynamics and omit its downstream evaluation in Table 4.

Experiment 1: k_1 as loss versus no KL across β. Figure 10 compares no-KL training against k_1 **as loss** at $\beta = 0.5$ and $\beta = 0.001$ on Qwen2.5-32B. The three curves stay close on reward, score, and format metrics, consistent with the 7B-scale finding (Figure 7). This again reflects the coefficient-level issue: changing the scalar weight on k_1 **as loss** does not introduce the missing reference-directed score-function coefficient.

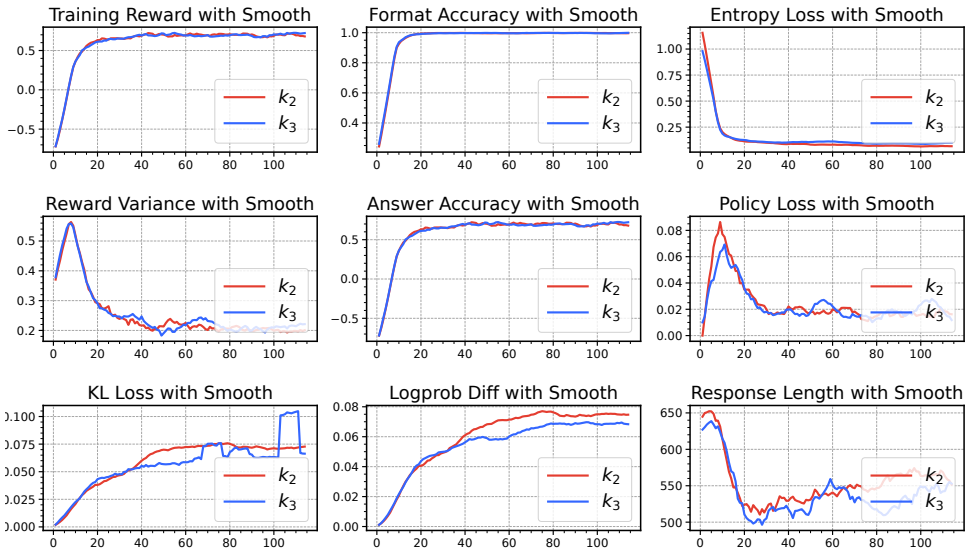

Figure 9: Qwen2.5-7B: k_2 **as loss** vs. k_3 **as loss** at the weaker $\beta = 0.001$. These 114-step, rollout-batch-64 runs are data-matched to the 228-step runs in total prompt count. The **KL Loss** panel shows a visible spike for k_3 **as loss** that is absent for k_2 **as loss**, matching the main-text weak-β pattern in Figure 3.

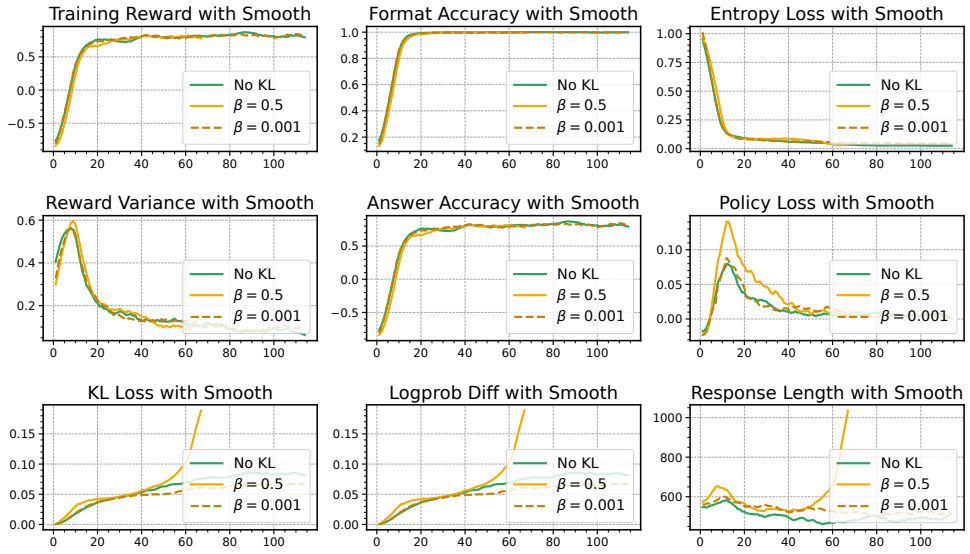

Figure 10: Qwen2.5-32B: no KL vs. k_1 **as loss** with $\beta \in \{0.5, 0.001\}$. The $\beta = 0.5$ run is partial (67 steps); the other two runs are complete (114 steps). Across the available training dynamics, k_1 **as loss** remains close to the no-KL baseline and does not impose an effective reference-coupling penalty.

Experiment 2: k_2 as loss versus k_3 as loss at $\beta = 0.5$. Figure 11 shows the $\beta = 0.5$ comparison on Qwen2.5-32B (114-step runs). The two KL heads instantiate the same coefficient contrast as at smaller scales, and in this setting k_2 **as loss** also slightly outperforms k_3 **as loss** on both math and general averages in Table 4.

Weaker KL weight ($\beta = 0.001$). Figure 12 shows the $\beta = 0.001$ comparison on Qwen2.5-32B. As in the 7B weak-β run, the reward-side metrics are close, but the **KL Loss** panel shows the same k_3 spike found at smaller scales.

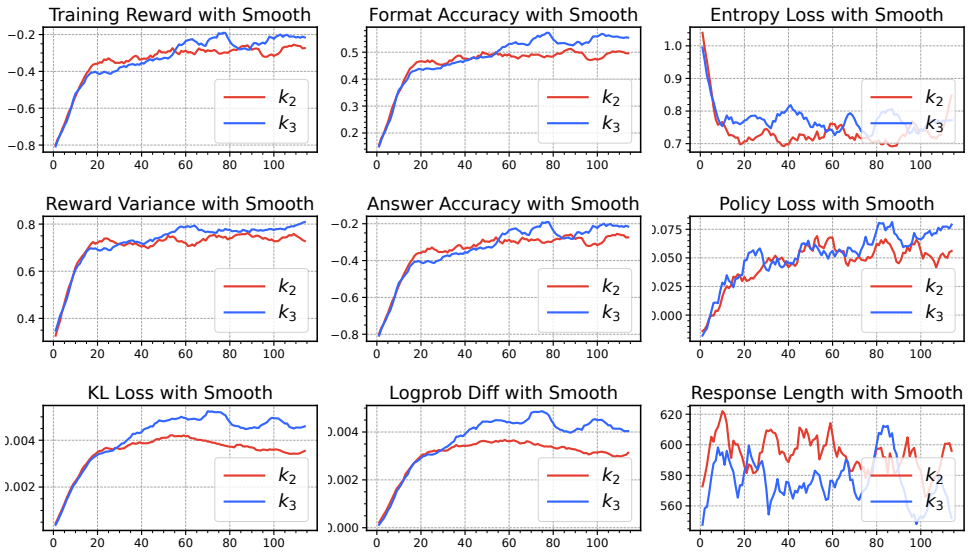

Figure 11: Qwen2.5-32B: k_2 **as loss** vs. k_3 **as loss** at $\beta = 0.5$ over 114 steps. The same nine training metrics compare the RKL-correct coefficient with the asymmetric-tail k_3 **as loss** surrogate; this setting also yields slightly higher downstream math and general averages for k_2 **as loss** than k_3 **as loss** in Table 4.

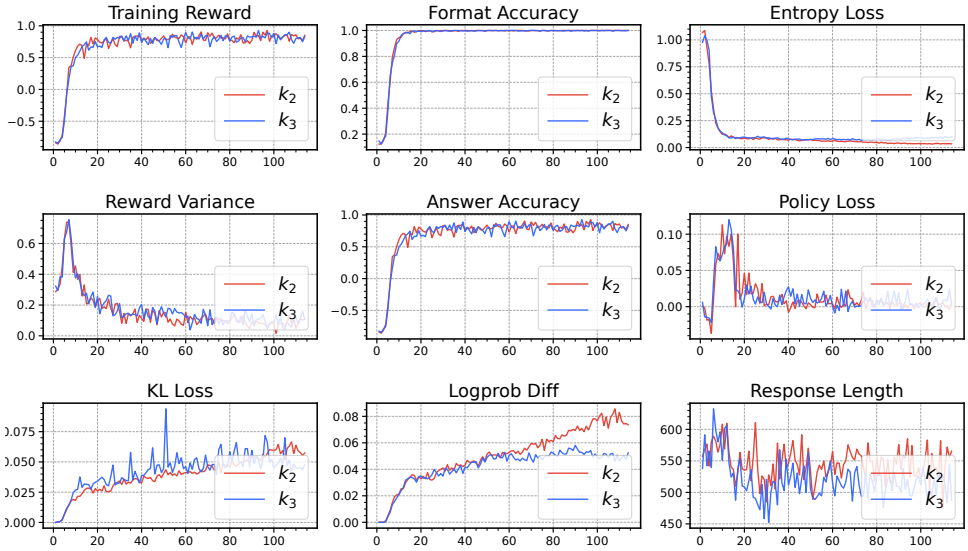

Figure 12: Qwen2.5-32B: k_2 **as loss** vs. k_3 **as loss** at the weaker $\beta = 0.001$ over 114 steps. The **KL Loss** panel shows a visible spike for k_3 **as loss** that is absent for k_2 **as loss**, showing the same weak-regularization pattern at 32B scale.

Summary. The standard-base-model experiments at 7B and 32B scale reproduce the patterns from the main text (Qwen2.5-Math-7B) and the 1.5B appendix (Qwen2.5-Math-1.5B): k_1 **as loss** does not provide meaningful regularization across the β values we tested; k_2 **as loss** implements the RKL-correct decoupled coefficient; and the k_3 surrogate's weak-β KL-loss spike persists even when the base checkpoint has not been math-specific post-trained. The 1.5B results show that score gaps can be small in low-drift regimes, while the 7B and 32B runs make the asymmetric-tail mismatch more visible under larger optimization pressure and scale.

Downstream performance. Table 4 reports downstream benchmark accuracy after training for both base models. The results echo the Qwen2.5-Math findings in Table 3: methods without effective KL constraints (no KL, k_1 **as loss**) retain the highest accuracy, consistent with less-constrained policy drift; effective KL heads trade accuracy for stronger reference-directed regularization, especially at the stronger $\beta = 0.5$. Thus, when k_2 **as loss** scores lower than no-KL or k_1 **as loss**, this is consistent with stronger regularization rather than worse implementation of RKL. Under the weaker $\beta = 0.001$, the KL variants land close to the no-KL baseline, and k_2 **as loss** matches or slightly exceeds k_3 **as loss** in the aggregate averages on both 7B and 32B. For 32B k_1 **as loss** at $\beta = 0.5$, training terminated after 67 steps due to a CUDA out-of-memory error (only 59% of the 114-step schedule), so the corresponding downstream evaluation was not run; the partial training dynamics are visible in Figure 10.

Table 4: Downstream benchmark accuracy (%) for RL training on the standard Qwen2.5-7B and Qwen2.5-32B base checkpoints. AIME reports AIME 24 / AIME 25. The "Math Avg." column averages AIME 24, AIME 25, AMC, MATH-500, Minerva, and Olympiad. Averages are computed from underlying unrounded scores. GPQA* denotes the GPQA evaluation setting used in our benchmark suite. The table is intended to complement the training metrics: tighter reference coupling can reduce task accuracy at large β, while weak-β runs remain close to the no-KL baseline.

Model	Math Reasoning Performance						General Domain Reasoning Performance			
	AIME 24/25	AMC	MATH-500	Minerva	Olympiad	Avg.	ARC-c	GPQA*	MMLU-Pro	Avg.
Qwen2.5-7B	7.8/4.1	27.6	44.0	10.3	23.0	19.5	15.1	2.0	21.1	12.7
RL w/o KL ($\beta = 0$)	14.1/8.8	45.3	76.2	36.4	40.7	36.9	89.2	32.8	55.0	59.0
RL w/ k_1 as loss, $\beta = 0.5$	12.2/6.4	44.5	75.0	35.7	40.4	35.7	89.5	36.4	54.3	60.1
RL w/ k_2 as loss, $\beta = 0.5$	9.1/4.8	34.2	56.6	13.6	30.1	24.7	36.8	4.5	26.7	22.7
RL w/ k_3 as loss, $\beta = 0.5$	8.2/5.3	32.3	56.8	12.9	27.1	23.8	37.2	5.1	26.7	23.0
RL w/ k_1 as loss, $\beta = 0.001$	13.6/6.7	45.6	75.6	34.6	41.9	36.3	89.5	36.9	54.6	60.3
RL w/ k_2 as loss, $\beta = 0.001$	11.8/5.4	44.7	77.4	36.4	40.0	35.9	88.9	36.9	54.1	60.0
RL w/ k_3 as loss, $\beta = 0.001$	12.7/4.3	43.5	74.2	36.4	41.5	35.4	89.8	31.8	54.1	58.5
Qwen2.5-32B	5.6/2.9	28.4	51.0	22.4	20.0	21.7	38.9	12.1	46.5	32.5
RL w/o KL ($\beta = 0$)	20.0/14.0	58.4	83.6	44.5	48.7	44.9	94.5	43.4	68.2	68.7
RL w/ k_1 as loss, $\beta = 0.5$ †	—	—	—	—	—	—	—	—	—	—
RL w/ k_2 as loss, $\beta = 0.5$	9.1/5.2	37.2	65.0	30.1	30.5	29.5	53.0	16.2	52.0	40.4
RL w/ k_3 as loss, $\beta = 0.5$	9.9/4.0	34.9	61.0	30.1	30.4	28.4	43.7	21.7	50.1	38.5
RL w/ k_1 as loss, $\beta = 0.001$	20.9/14.4	58.3	83.6	43.4	46.8	44.6	94.1	48.5	67.7	70.1
RL w/ k_2 as loss, $\beta = 0.001$	19.8/13.8	56.9	85.0	42.6	49.2	44.6	93.9	43.4	67.8	68.4
RL w/ k_3 as loss, $\beta = 0.001$	18.0/13.9	56.4	84.0	42.6	47.7	43.8	94.0	45.5	64.5	68.0

† Training terminated after 67 steps due to a CUDA out-of-memory error; no downstream evaluation was run.

Q Note on Closely Related Discussions

We are aware of closely related subsequent and contemporaneous discussions of KL-gradient pitfalls in RLHF, including Tang & Munos (2025), Shah et al. (2025), and the online discussion by Wang (2025). These works discuss issues very close to the ones studied here, including differentiating through single-sample KL estimates and the distinction between KL value estimation and KL-gradient correctness. We cite them to make clear that such related discussions exist, but surface-level overlap alone is not a sufficient basis for assessing originality: the relevant question is the detailed chronology of independent development, circulation, and public release. Because this submission is under double-blind review, we do not use this appendix to assert a public priority timeline or to disclose non-anonymous circulation details. If reviewers or area chairs consider originality relative to these discussions central to the evaluation, we respectfully ask that they first conduct a careful chronology-based investigation; we can provide the relevant timeline through the appropriate confidential review channel or after the anonymous review phase.

R Limitations

Although we evaluate multiple model scales and KL settings, compute constraints prevent us from testing hundred-billion- or trillion-parameter models. We also do not report error bars or formal

significance tests for the RLHF runs. Our training data is primarily mathematical, so we do not systematically study other subject domains or non-math reasoning tasks.

S BROADER IMPACT STATEMENT

This work is methodological and aims to improve the stability and reproducibility of RLHF training by clarifying the effects of KL regularization on gradients. Better-grounded KL implementations can reduce training instability and wasted compute, and may lower the risk of unintended behaviors caused by uncontrolled policy drift during alignment. At the same time, improvements to RLHF optimization can be used in both beneficial and harmful applications; we encourage practitioners to evaluate downstream impacts and follow responsible deployment practices.