

A Comedy of Estimators: On KL Regularization in RL Training of LLMs

Vedant Shah^{* 1,2}, Johan Obando-Ceron^{* 1,2}, Vineet Jain^{* 1,3}

Brian R. Bartoldson⁵, Bhavya Kailkhura⁵, Sarthak Mittal^{1,2}, Glen Berseth^{1,2,7}

Pablo Samuel Castro^{1,2}, Yoshua Bengio^{1,2,4,7}, Esmeralda S. Whitammer^{6,8}

Moksh Jain^{† 1,2}, Siddarth Venkatraman^{† 1,2}, Aaron Courville^{† 1,2,7}

^{*}Equal contribution [†]Equal senior contribution

¹Mila – Québec AI Institute ²Université de Montréal ³McGill University

⁴LawZero ⁵LLNL ⁶University of Edinburgh ⁷CIFAR AI Chair ⁸CIFAR Fellow

{vedant.shah, johan.ceron, jain.vineet, mittalsa, glen.berseth,
pablo-samuel.castro, yoshua.bengio, moksh.jain,
siddarth.venkatraman, courvila}@mila.quebec

{bartoldson1, kailkhura1}@llnl.gov, esmeralda.whitammer@ed.ac.uk

Abstract

The reverse Kullback-Leibler (KL) divergence between the trained policy and a reference policy is commonly used as a regularizer when training large language models (LLMs) with reinforcement learning (RL). Since computing the exact sequence-level KL divergence is intractable, practical algorithms use sample-based estimators computed from on-policy rollouts and incorporate them in the reward or the loss. Despite its ubiquity, the interaction between estimator choice, position of the regularization, and downstream performance is poorly understood. Recent work also identifies that certain implementation choices can result in biased gradients. We analyze these practices and examine the gradients of several estimator configurations, showing how the choices shape gradient bias. We substantiate these observations with empirical evidence by RL fine-tuning LLMs with different KL configurations, and evaluating their performance on both in- and out-of-distribution tasks. In on-policy settings, we find that biased-gradient estimator configurations can show training instabilities, while unbiased-gradient configurations lead to better performance on in-domain as well as out-of-domain tasks. We also observe that KL regularization can help stabilize asynchronous training. Overall, our findings provide useful takeaways for KL-regularized objectives during RL post-training of LLMs.

1 Introduction

Reinforcement learning (RL) is a key component of post-training pipelines for large language models (LLMs). [Jaech et al. \(2024\)](#); [Guo et al. \(2025\)](#) have shown that training LLMs with RL on verifiable reasoning tasks such as math and coding improves reasoning abilities. This has precipitated rapid development in reasoning-oriented RL approaches for LLMs ([Lambert, 2025](#)). At the same time, this has resulted in inconsistent design choices and implementations in open-source RL fine-tuning pipelines ([Tang & Munos, 2025](#); [Zhang et al., 2025](#)).

One such design choice is the use of Kullback-Leibler (KL) divergence between the trained policy and a reference policy as a regularization term in the objective ([Ziegler et al., 2019](#); [Shao et al., 2024](#)). It is designed to constrain the policy within the support of the reference model, mitigating problems such as language drift, reward over-optimization ([Gao et al., 2023](#)), and catastrophic forgetting ([Qi et al., 2024](#)). Traditionally, the *reverse* KL divergence is used for this regularization, with a regularization coefficient controlling the trade-off between reward maximization and proximity to the reference model. However, exact

computation of the reverse KL divergence is intractable due to the combinatorially large space of possible sequences. As a result, different sample-based estimators of the reverse KL divergence are used in practice (Schulman, 2020; Amini et al., 2025).

In addition to differences in their approximations, these estimators may be incorporated into the objective in different ways: RLHF with PPO (Ouyang et al., 2022) adds the KL penalty to the task reward (*i.e.*, no direct gradients); methods such as GRPO (Shao et al., 2024; Guo et al., 2025) popularized adding it directly to the loss. The choice of the estimator, regularization coefficient, and whether it is added to the reward or directly to the loss can have a significant effect on the training stability, convergence rate, and out-of-distribution generalization of the trained models. Moreover, previous work (Tang & Munos, 2025; Zhang et al., 2025) has identified that some of these choices lead to biased estimates of the true gradient. For example, using the KL estimator in the loss function, as popularized by GRPO, results in biased gradients and therefore does not optimize the intended reverse KL-regularized objective.

While KL regularization is ubiquitous in RL training of LLMs and theoretically well-understood, there is a lack of empirical investigation of the effects of these choices on the downstream performance. **In this work, we fill this gap by providing a systematic analysis and empirical investigation of the space of these design choices, with respect to sequence-level reverse KL regularization as the intended objective.** We study this in the context of reinforcement learning with verifiable rewards (RLVR; Trung et al., 2024; Lambert et al., 2025), which has become the dominant paradigm for improving the reasoning abilities of LLMs. Specifically, we investigate two commonly used *unbiased* estimators of reverse KL divergence in on-policy settings – the naïve or **K1** estimator, and the Schulman or **K3** estimator (Schulman, 2020). These are the most common estimators used in practice, hence we limit our scope to these estimators and do not include others, such as the squared or **K2** estimator, in our analysis.

First, we give an overview of the bias of the gradients induced by different estimator configurations with respect to the true gradient of *sequence-level* reverse KL divergence. While both estimators are unbiased, they do not necessarily induce an unbiased gradient update (§2.1, table 1). We demonstrate this empirically in a minimal setting where computing the true gradient of sequence-level reverse KL divergence is tractable (§3.3). Next, we empirically study the implications of these choices in RL fine-tuning of Qwen2.5-7B (Qwen et al., 2024) and Llama-3.1-8B-Instruct (Touvron et al., 2023) on a mathematical reasoning task in an on-policy setting. We also study both in- and out-of-domain performance of the resulting models (§4). Finally, we conduct a similar study in a highly asynchronous RL training setup (Noukhovitch et al., 2025; Bartoldson et al., 2025) by training Qwen2.5-7B on a mathematical reasoning task and Qwen3-4B-Instruct-2507 (Yang et al., 2025) on a general reasoning task.

Key observations:

- Unbiased estimates of the sequence-level reverse KL divergence can result in *biased* gradients depending on their usage.
- Configurations inducing biased gradients can lead to unstable training and even precipitate collapse.
- Configurations that lead to unbiased gradient estimates result in better-performing models, on both in- and out-of-domain evaluation tasks.
- KL regularization can stabilize learning from highly off-policy data in asynchronous setups.

2 Background

We study the problem of fine-tuning a language model π_θ with reinforcement learning. Given a reward function $R(\cdot)$ and a set of observations $\mathcal{D}$ comprising question-answer pairs (x, y) , RL fine-tuning of LLMs optimizes the following objective

$$\max_{\theta} \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} [R(y_{1:T}, y)] - \beta \text{KL}(\pi_\theta(\cdot | x) \parallel \pi_{\text{ref}}(\cdot | x)) \right], \quad (1)$$

where β is a hyperparameter that controls the weight of the KL divergence penalty, π_{ref} is the reference policy, and $y_{1:T}$ denotes solutions generated by the model conditioned on the question. Since both the sampling of $y_{1:T}$ and R are non-differentiable, the objective is optimized using policy gradient methods such as PPO (Ouyang et al., 2022) or GRPO (Shao et al., 2024).

We study the reinforcement learning with verifiable rewards (RLVR) setting where R is an oracle verifier. An example is a reward function that checks the generated answer against the ground truth answer: $R(y_{1:T}, y) = \mathbb{1}_{\text{Extract}(y_{1:T})=y}$, where $\text{Extract}(\cdot)$ pulls the solution from a `\boxed{\{}}` block within the solution. In general, R could be any reward model with y specifying the data required for reward computation.

The objective in Eq. (1) incentivizes the policy π_θ to maximize the expected reward while remaining close to a reference model π_{ref} . This constraint is enforced via a *reverse* KL divergence penalty in the objective, which is an expectation under the learned policy π_θ :

$$\text{KL}(\pi_\theta(\cdot | x) \| \pi_{\text{ref}}(\cdot | x)) = \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} \left[\log \frac{\pi_\theta(y_{1:T} | x)}{\pi_{\text{ref}}(y_{1:T} | x)} \right] \quad (2)$$

Additionally, a control variate in the form of a *baseline* $b(x, y)$ is subtracted from the reward to reduce variance during training, resulting in the *advantage* $A(x, y, y_{1:T}) = R(x, y, y_{1:T}) - b(x, y)$ ¹.

2.1 Unbiased Gradient of a KL Estimator

In this work, we are interested in estimating the reverse KL divergence between the *sequence-level distributions* $\pi_\theta(y_{1:T} | x)$ and $\pi_{\text{ref}}(y_{1:T} | x)$. It is commonly estimated by decomposing into token-level estimates (Sheng et al., 2025; von Werra et al., 2020; Hu et al., 2024; Cui et al., 2026). For a generic sequence-level reverse KL divergence estimator $\widehat{\text{KL}}$,

$$D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) = \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} [\widehat{\text{KL}}] = \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} \left[\sum_{t=1}^T \widehat{\text{KL}}_t \right], \quad (3)$$

where $\widehat{\text{KL}}_t$ is the estimator defined on the *token-level distributions* $\pi_\theta(y_t | x, y_{<t})$ and $\pi_{\text{ref}}(y_t | x, y_{<t})$. Henceforth, any reference to an estimator implies reference to the use of the token-level version $\widehat{\text{KL}}_t$. The gradient of the expectation of $\widehat{\text{KL}}$, under sequences sampled from π_θ can then be written as

$$\nabla_\theta \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} [\widehat{\text{KL}}] = \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} \left[\sum_{t=1}^T \nabla_\theta \widehat{\text{KL}}_t + \sum_{t=1}^T \widehat{\text{KL}}_t \nabla_\theta \log \pi_\theta(y_{1:T} | x) \right]. \quad (4)$$

Such gradient estimators have been previously studied in Ranganath et al. (2014); Tang & Munos (2025); Zhang et al. (2025). See App. F.2 for a derivation of Eq. (4). Next, we discuss two ways of using the estimator in the context of RL training of LLMs. We restrict our primary analysis in this section to the on-policy setting.

2.2 Placement of the KL Estimator

We now discuss the different ways KL estimators have been used in the RL objective – namely, adding the estimator to the reward, and adding it directly to the loss objective.

Reward. An estimator is added to the reward by applying a stop-gradient operation on the KL estimate and adding it to the token-level task score. The gradient corresponding to the KL penalty is then computed as $\mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} \left[- \sum_{t=1}^T \widehat{\text{KL}}_t \cdot \nabla_\theta \log \pi_\theta(y_{1:T} | x) \right]$.

¹The baseline is chosen to have an expected value of 0 under the policy and does not affect the optima.

Table 1: **Summary of estimators considered in this study and the bias of their gradients with respect to the true gradient of sequence-level reverse KL divergence in on-policy settings.** We study 4 settings, including the commonly used **K1** estimator in reward and **K3** estimator in loss. All configurations except using **K1**-in-reward lead to biased gradients. In two of the cases of biased gradients, we observe training instabilities or collapses when they are used in RL fine-tuning of LLMs. $r_t = \frac{\pi_{\text{ref}}(y_t | x_t, y_{<t})}{\pi_{\theta}(y_t | x_t, y_{<t})}$ in the expressions given.

Estimator	Expression	Position	Unbiased Grad. Est. (§2.1)	Training (§4)
K1	$-\log r_t$	Reward	✓	Stable
		Loss	✗	Unstable
K3	$r_t - 1 - \log r_t$	Reward	✗	Collapse
		Loss	✗	Stable

Loss. This refers to adding the KL estimator to the loss, as popularized by GRPO (Guo et al., 2025; Shao et al., 2024). Automatic differentiation, which is commonly used in practice, cannot backpropagate through the sampling process used to compute the KL estimate. The gradient of the objective is then computed as $\mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot | x)} \left[-\sum_{t=1}^T \nabla_{\theta} \widehat{\text{KL}}_t \right]$.

We refer the reader to Appendix F.1 and F.2 for a derivation of the expressions discussed in this section. Note that in the general case, both of these terms in isolation are biased with respect to the correct gradient as stated in Eq. (4). In the case of on-policy training, we can recover the correct gradient up to a normalizing factor by adding the estimator to both the reward and the loss.

3 Inspecting KL Estimators

In principle, if $\widehat{\text{KL}}$ is an estimator for the reverse KL divergence in the sense of Eq. (3), then it should satisfy:

$$\nabla_{\theta} \mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot | x)} [\widehat{\text{KL}}] = \mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot | x)} \left[\log \frac{\pi_{\theta}(y_{1:T} | x)}{\pi_{\text{ref}}(y_{1:T} | x)} \nabla_{\theta} \log \pi_{\theta}(y_{1:T} | x) \right], \quad (5)$$

where the left side is given by Eq. (4) and the right side is the gradient of the true sequence-level reverse KL divergence.

We now examine two commonly used unbiased KL estimators used in RL training of LLMs, namely the naïve or **K1** estimator, and the Schulman or **K3** estimator (Schulman, 2020), relative to the true gradient in Eq. (5). While the K2 or squared estimator is another known KL estimator, we restrict the scope of this paper to unbiased and popularly used estimators. We refer the reader to Appendix F.3 and F.4 for detailed derivations of the gradient expressions mentioned in this section. Throughout the discussion, $r_t = \frac{\pi_{\text{ref}}(y_t | x_t, y_{<t})}{\pi_{\theta}(y_t | x_t, y_{<t})}$.

3.1 K1 Estimator

The **K1** estimator is the Monte Carlo estimate of the log-ratio of likelihoods under the current and reference policies, with samples from the current training policy. Given below are the expressions of the estimator and the gradients induced by its use in reward and loss. The gradient estimate induced when **K1** is used in the reward is unbiased, whereas it is biased when the estimator is used in loss.

$$\mathbf{K1} = \sum_{t=1}^T \mathbf{K1}_t = \sum_{t=1}^T \log \frac{\pi_{\theta}(y_t | x_t, y_{<t})}{\pi_{\text{ref}}(y_t | x_t, y_{<t})} = - \sum_{t=1}^T \log r_t. \quad (6)$$

Reward.

$$\mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot|x)} \left[\sum_{t=1}^T \mathbf{K1}_t \nabla_\theta \log \pi_\theta(y_{1:T} | x) \right] = \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot|x)} \left[- \sum_{t=1}^T \log r_t \cdot \nabla_\theta \log \pi_\theta(y_{1:T} | x) \right]. \quad (7)$$

Loss.

$$\mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot|x)} \left[\sum_{t=1}^T \nabla_\theta \mathbf{K1}_t \right] = \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot|x)} \left[\nabla_\theta \log \frac{\pi_\theta(y_{1:T} | x)}{\pi_{\text{ref}}(y_{1:T} | x)} \right] = 0. \quad (8)$$

$\mathbf{K1}$ is a special case where the gradient of the expectation of the estimator obtained from adding KL estimator in the reward results in an unbiased estimator (Tang & Munos, 2025). This is because the first term of equation Eq. (4) is zero in expectation for the $\mathbf{K1}$ estimator as shown in Eq. (8).

3.2 K3 estimator

The $\mathbf{K3}$ has a lower variance and thus is often preferred over $\mathbf{K1}$ in practice. $\mathbf{K3}$ induces biased gradient estimates both when used in reward or loss. The estimator and gradient expressions are as given below.

$$\mathbf{K3} = \sum_{t=1}^T \mathbf{K3}_t = \sum_{t=1}^T \left[\frac{\pi_{\text{ref}}(y_t | x, y_{<t})}{\pi_\theta(y_t | x, y_{<t})} - 1 - \log \frac{\pi_{\text{ref}}(y_t | x, y_{<t})}{\pi_\theta(y_t | x, y_{<t})} \right] = \sum_{t=1}^T [r_t - 1 - \log r_t]. \quad (9)$$

Reward.

$$\nabla_\theta \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot|x)} \left[\sum_{t=1}^T \mathbf{K3}_t \nabla_\theta \log \pi_\theta(y_{1:T} | x) \right] = \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot|x)} \left[\sum_{t=1}^T (r_t - \log r_t) \cdot \nabla_\theta \log \pi_\theta(y_{1:T} | x) \right]. \quad (10)$$

Loss.

$$\mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot|x)} \left[\sum_{t=1}^T \nabla_\theta \mathbf{K3}_t \right] = \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot|x)} \left[- \sum_{t=1}^T r_t \cdot \nabla_\theta \log \pi_\theta(y_t | x, y_{<t}) \right]. \quad (11)$$

In particular, the gradient of the expectation of the KL estimate when $\mathbf{K3}$ is used in the loss is a biased estimate of the true reverse KL gradient shown in Eq. (4). This is the version used in the implementations of some of the most popular RL algorithms for LLMs, such as GRPO (Shao et al., 2024; Guo et al., 2025).

Off-policy training. In the off-policy setting, none of the KL configurations we discuss are unbiased due to the difference in the sampling and training policy. Zhang et al. (2025) note that an unbiased gradient estimate of the *token-level* reverse KL gradient can be computed as the token-level importance sampling ratio ω_t weighted token-level KL estimate added to the loss directly (i.e., $\omega_t \mathbf{K3}_t$ or $\omega_t \mathbf{K1}_t$). However, we focus on *sequence-level* reverse KL objective in this work. The sequence-level reverse KL objective implementations resulting in unbiased reverse-KL estimates are susceptible to high variance from the sequence-level importance sampling ratio ω_s for long sequences. In practice, the sequence-level KL objective is weighed by the token-level importance sampling ratios when KL is used in the reward, and the importance ratio is skipped entirely when used in the reward (as in GRPO). Thus, for the off-policy setting, we primarily study the empirical performance of the KL configurations.

3.3 Parametric Autoregressive Model: An Illustrative Example

We first study the bias in the gradients of the KL estimators in a minimal parametric autoregressive model. We define reference models A and B over binary sequences, each

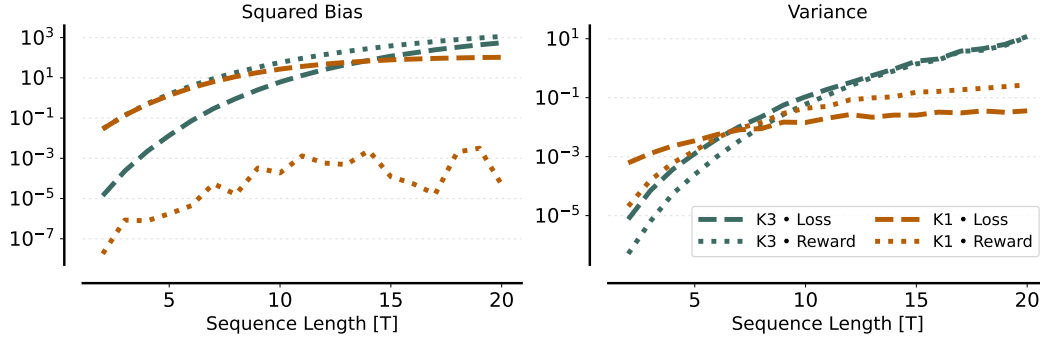


Figure 1: The bias and variance of expected gradients with respect to the parameters of A , in different configurations (logarithmic scale). While all estimators are unbiased, the expected gradients are unbiased only in the case of **K1** estimator when used in reward. **K3** estimator when used in reward exhibits the highest gradient bias. While **K1** estimator in loss has relatively lower variance in the estimated gradient, it also suffers from high bias.

factorizing into Bernoulli conditionals over each token in the sequence (conditioned on the previous tokens).

$$\begin{aligned}
 A_\theta(Y) &= \prod_{t=1}^T (p_t^A)^{y_t} (1 - p_t^A)^{1-y_t}, p_t^A = \sigma(a + b c_{t-1}), \quad c_{t-1} = \sum_{k=1}^{t-1} y_k. \\
 B_\phi(Y) &= \prod_{t=1}^T (p_t^B)^{y_t} (1 - p_t^B)^{1-y_t}, p_t^B = \sigma(\tilde{a} + \tilde{b} c_{t-1}), \quad c_{t-1} = \sum_{k=1}^{t-1} y_k.
 \end{aligned} \tag{12}$$

Here, Y is a binary sequence, $a, b, \tilde{a}$ and $\tilde{b}$ are the parameters of distributions A and B , $c_0 = 0$, $y_t \in \{0, 1\}$ and σ represents the sigmoid function. The reverse KL $D_{\text{KL}}(A \parallel B)$ and its gradient with respect to a and b admit closed-form expressions (see App. C).

We compute the KL divergence with different estimators and their gradients when used as reward and in the loss as discussed in §2.1, using 200 trials each with $N = 1000$ sequences of lengths T sampled from A . We illustrate the bias and variance of the gradient estimates of the different configurations in Fig. 1. Note the logarithmic scale of the plots. We observe that the bias of the gradient of the **K1** estimator added to the reward remains low. On the other hand, the gradients associated with the **K3** estimator in the loss and the reward both show high bias and variance. These results validate the conclusions from §2.1.

4 RL Fine-tuning of LLMs

We complement our analysis in §3 with an empirical study on the effect of design choices in KL regularization on training LLMs with RL. We use the initial base model as the reference model π_{ref} in all experiments. We conduct the following investigations:

- (§4.1) How do the various KL configurations discussed in §3 affect the downstream performance of LLMs trained using on-policy RL in both in- and out-of-domain tasks?
- (§4.2) How do different KL configurations affect the downstream performance of models when used in an asynchronous RL (i.e. off-policy) setup.

Motivated by our observations in this section, we list the recommended settings to induce the unbiased KL gradient estimate for commonly used libraries in App. E.

4.1 Effect of KL Configurations in On-policy RL

Experimental Setup. We (on-policy) RL fine-tune Qwen2.5-7B and Llama-3.1-8B-Instruct models on the training subset of Hendrycks MATH (Hendrycks et al., 2021) (henceforth referred to as MATH) with different KL configurations, across different values of the β coefficient. We evaluate the trained models on 2 different in-domain tasks, namely

MATH500 (Lightman et al., 2023) and MATH² (Shah et al., 2024) and 3 out-of-domain tasks – MMLU college physics, college chemistry and college biology subsets (Hendrycks et al., 2021). To demonstrate training behavior, we report the train-time Pass@1 score of the models on an in-distribution but held-out test set of the task and report mean@32 accuracy across 3 seeds on the final evaluations. Models are evaluated at the same sampling temperature they are trained at. More details about the experimental setup are discussed in App. B. We also validate our observations on an additional experimental setup involving Qwen3-4B-Instruct-2507 (Yang et al., 2025) and CodeIO (Stojanovski et al., 2025) in App. D.3.

In order to restrict our study on the effect of the KL estimators, we opt to use REINFORCE leave-one-out advantage baseline (Ahmadian et al., 2024, RLOO). We use a batch size of 256, maximum response length 1024 tokens, a learning rate of 10^{-6} and sample $K = 5$ responses per prompt at a temperature of 1.0. We discuss our main observations below.

Observation 1: Adding K_1 estimator to the loss can lead to training instabilities.

As shown in Eq. (8), adding K_1 to the loss results in a biased estimate of the reverse KL gradient, since the term is zero in expectation. Intuitively, RL fine-tuning with K_1 with any coefficient β should perform similar to RL fine-tuning without any KL penalty ($\beta = 0$). To verify this empirically, we fine-tune Qwen2.5-7B (Qwen et al., 2024) and Llama-3.1-8B-Instruct (Touvron et al., 2023) with $\beta = 0.05, 0.1, 0.3$ and 1 and compare them against RL fine-tuning with $\beta = 0$. Fig. 2 shows Pass@1 performance of models on MATH test set over the course of training.

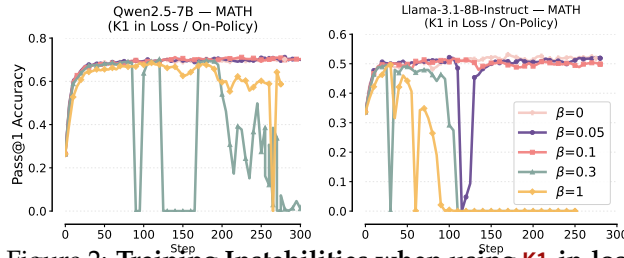


Figure 2: **Training Instabilities when using K_1 -in-loss.** Pass@1 performance for [Left] training Qwen2.5-7B with K_1 , leads to training instabilities for $\beta = 0.1$ and 1. [Right] Training Llama-3.1-8B-Instruct with K_1 in loss leads to instabilities for all β except 0.1.

We observe training instabilities across both Qwen2.5-7B and Llama-3.1-8B-Instruct models, and different values of β as shown in Fig. 2. A potential explanation is that the term $\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(y_t | x, y_{<t})$, despite having an expectation of 0, adds variance to the optimization, leading to instabilities. In cases where the training is stable, i.e., Qwen2.5-7B trained with $\beta = 0.05$ and 0.1, the performance is similar to training without any KL, as expected. Qwen2.5-7B seems to be more robust to variance as compared to Llama-3.1-8B-Instruct models.

Observation 2: Adding K_3 estimator to the reward leads to training collapse.

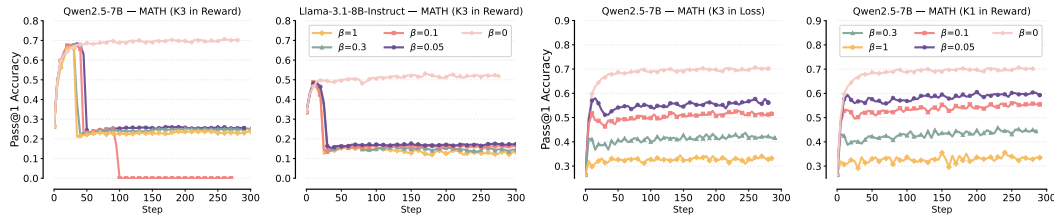


Figure 3: **Collapse in the case of adding K_3 to the reward.** First: Pass@1 performance for Qwen2.5-7B trained on MATH train dataset. Second: Llama-3.1-8B-Instruct trained on MATH train dataset with K_3 -in-reward. The collapse may be attributed to high bias and variance of the configuration. Pass@1 on MATH test set with K_3 -in-loss (biased; third) and K_1 -in-reward (unbiased; fourth). Although biased with respect to reverse KL, K_3 -in-loss yields stable training. In both cases, lower β values lead to higher performance.

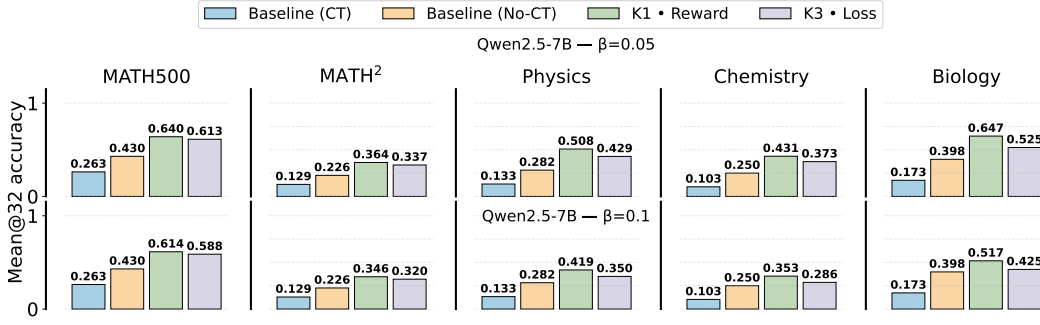


Figure 4: Comparison of Qwen2.5-7B trained with two stable estimator configurations - **K1-in-reward** and **K3-in-loss**. Baseline (CT) refers to the performance of base Qwen2.5-7B when prompted with a chat template. Baseline (No-CT) represents the performance when it is prompted without a chat template. **K1-in-reward** (unbiased gradient) performs the best on both in-domain and out-of-domain tasks. Increasing β deteriorates performance.

Another case of biased gradient estimate is **K3** used in the reward. From Eq. (10), we observe that adding token-level **K3** to rewards leads to a bias term of

$$\mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot | x)} \left[\sum_{t=1}^T \frac{\pi_{\text{ref}}(y_t | x, y_{<t})}{\pi_{\theta}(y_t | x, y_{<t})} \nabla_{\theta} \log \pi_{\theta}(y_{1:T} | x) \right]. \quad (13)$$

This bias is illustrated in Fig. 1 for the parametric autoregressive model §3.3. To validate this in the case of LLMs, we RL fine-tune Qwen2.5-7B and Llama-3.1-8B-Instruct with $\beta = 0.05, 0.1, 0.3$ and 1. Fig. 3 (first and second) shows that this biased gradient estimate results in unpredictable behavior leading to complete or partial collapse for all β .

Observation 3: Unbiased gradient estimator results in better out-of-distribution performance as compared to biased estimators with stable training behaviors.

The final setting leading to a biased expected gradient of the sequence-level reverse KL is when **K3** is used in the loss Eq. (11). Surprisingly, despite the bias, using **K3** in the loss exhibits stable training of the policy. Fig. 3 (third) reports the in-distribution performance of Qwen2.5-7B models (evaluated on MATH test dataset) for different β during training. This may be explained by the observation that the gradient estimate Eq. (11) in this case is a sum of unbiased gradient estimates of the forward KL divergences computed at the token level, making this configuration equivalent to minimizing the forward KL divergence for each token (see App. F.4). Note that using **K3-in-loss** is a popular configuration used with policy optimization algorithms such as GRPO (Shao et al., 2024; Guo et al., 2025).

K1-in-reward, resulting in an unbiased estimate of the true gradient (Eq. (7)), demonstrates stable training as shown in Fig. 3 (fourth). We compare the downstream performance of the models trained with **K3-in-loss** and **K1-in-reward**. We train Qwen2.5-7B using these two estimator configurations for 250 steps on MATH train data, and compare the performance of the two configurations on a wide range of evaluation tasks as shown in Fig. 4. Similar results for Qwen2.5-7B with $\beta = 0.3$ and $\beta = 1$ can be found in App. D.1.

We observe that using **K1-in-reward** (i.e. unbiased gradient estimate) outperforms using **K3-in-loss** (i.e. biased gradient estimate). While the performance gains are consistent across all tasks and both models, we observe that the gains are more pronounced in out-of-domain tasks for Qwen2.5-7B, with an average relative improvement of 19.06% across MMLU college-physics, college-chemistry, and college-biology, as compared to an average relative improvement of only 6.21% on in-domain tasks, for $\beta = 0.05$. On the other hand, the gains are more pronounced in-domain (average relative improvement of 15.94%) than out-of-domain (average relative improvement of only 3.65%) for Llama-3.1-8B-Instruct.

We investigate the effect of using the KL term in both the reward and the loss (resulting in unbiased gradient estimates for both **K1** and **K3**, §2.2) in App. D.2, and we perform further analysis for **K3-in-loss** and **K1-in-reward** in App. D.5.

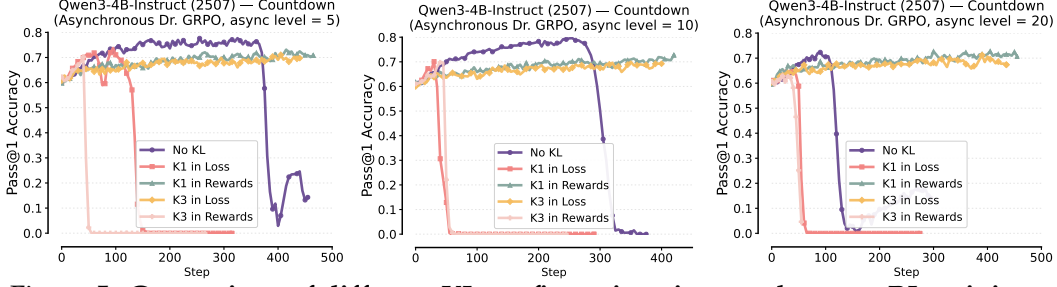


Figure 5: **Comparison of different KL configurations in asynchronous RL training of Qwen3-4B-Instruct-2507 on Countdown (async levels = 5, 10, 20).** Using **K1**-in-reward and **K3**-in-loss show stable training as opposed to using no KL regularization, **K1**-in-loss and **K3**-in-reward.

4.2 Effect of KL Configurations in Asynchronous RL

While all four KL configurations discussed result in biased sequence-level KL gradient updates in off-policy settings (as noted in §3), we study the performance of the four configurations empirically. We use an asynchronous RL setup (Noukhovitch et al., 2025; Bartoldson et al., 2025) which is commonly used for large-scale RL training runs in order to reduce latency. The asynchronous policy updates give rise to off-policyness.

Experimental Setup. We train Qwen3-4B-Instruct-2507 (Yang et al., 2025) on Countdown (Stojanovski et al., 2025), with Dr. GRPO (Liu et al., 2025) at a high asynchrony levels of 5, 10 and 20. We use a learning rate of 10^{-6} , sample 16 rollouts per prompt at a temperature of 1.0, use a training batch size of 512, maximum response length of 4096, and train for up to 350 steps, with $\beta = 0.005$. The rest of the experimental setup remains the same as in the on-policy experiments. Fig. 5 shows the test performance across different KL configurations for all async levels, during the course of training. The plots show the performance on a held-out set of countdown.

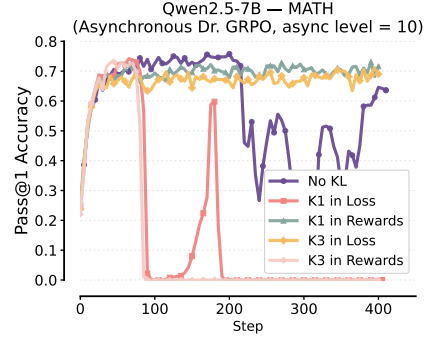


Figure 6: **Comparison of different KL configurations in asynchronous RL training of Qwen3-2.5-7B on Hendrycks MATH (async level = 10).** Staying consistent with the observations in async training of Qwen3-4B-Instruct-2507 + Countdown, using **K1**-in-reward and **K3**-in-loss show stable training as opposed to using no KL regularization, **K1**-in-loss and **K3**-in-reward.

Further, we also train Qwen2.5-7B on MATH with the same experimental setup and an async level of 10. Fig. 6 shows the performance on the test set of Hendrycks MATH over the course of the training. Fig. 7 shows the performance of the final checkpoints fine-tuned with **K1**-in-reward, and **K3**-in-loss (the two stable settings) on other in-domain and out-of-domain evaluation sets. Note that the performance gap between no KL training and KL regularized training can be closed by using regular reference policy resets as noted in Bartoldson et al. (2025).

Our observations are twofold:

- In highly asynchronous settings, **K1**-in-reward and **K3**-in-loss, both help stabilize the training in cases where it is otherwise unstable in the absence of any KL regularization.
- Staying consistent with observations in the case of on-policy RL, the use of **K1**-in-loss and **K3**-in-reward show training instabilities whereas the use of **K1**-in-reward and **K3**-in-loss show stable training. In evaluation, **K1**-in-reward outperforms **K3**-in-loss.

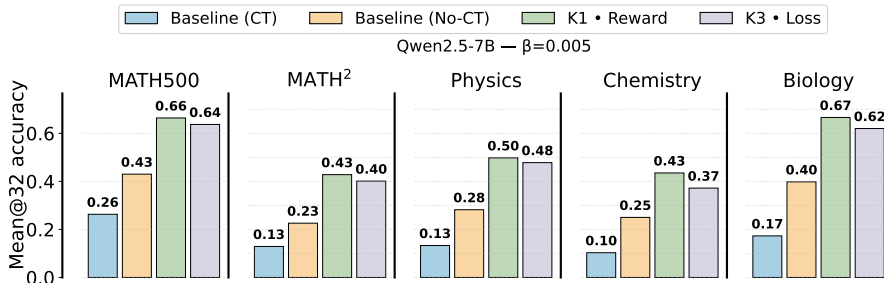


Figure 7: **Comparison of Qwen2.5-7B trained on MATH dataset in asynchronous setting with the two stable configurations - K1-in-reward and K3-in-loss.** Consistent with the observations in synchronous settings, K1-in-reward outperforms K3-in-loss across all considered evaluation tasks.

5 Conclusion

We conduct a study of how different KL estimators, and their placement within the RL objective, affect the stability and performance of RL fine-tuning of LLMs. We consistently find that implementations with biased reverse KL divergence gradient estimates perform unpredictably: at worst leading to training collapses and at best still underperforming implementations with unbiased gradient estimates. While the K3 estimator in the loss, commonly used in GRPO, remains generally stable, it consistently underperforms the naïve K1 in reward configuration. These results all suggest that unbiased gradient configurations should serve as the default for stable and generalizable RL post-training. Updating parameters with a step that is not the gradient of the desired objective will produce undesired optima; furthermore, making steps along a non-conservative vector field is a recipe for instability, as stable behavior near an optimum is not guaranteed. We hope that our systematic study will help clarify these issues for the community.

Ethics Statement

Our work focuses on studying the effect of various design choices for KL regularization in training language models with RL on the performance on downstream tasks. While we foresee no immediate negative societal consequences of our work, we hope that future researchers who build upon it will bear in mind the potential of LLMs – and in particular of human-like reasoning in LLMs – to be used both for good and for harm.

Acknowledgments

The authors thank Anirudh Goyal, Michael Noukhovitch and Emiliano Penaloza for interesting discussions and helpful comments. The research was enabled in part by computational resources provided by the Digital Research Alliance of Canada (<https://alliancecan.ca>), Mila (<https://mila.quebec>), NVIDIA, and the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award ASCR-ERCAP0032802 and NERSC award ASCR-ERCAP0032812.

AC and YB acknowledge funding from National Sciences and Engineering Council of Canada (NSERC) and the Canadian Institute for Advanced Research (CIFAR). AC acknowledges funding from the AI for Math fund (<https://www.renaissancephilanthropy.org/ai-for-math-fund>) and Google, supporting the VS and JO graduate programs. GB acknowledges funding from NSERC and CIFAR. ESW acknowledges support from the CIFAR Learning in Machines and Brains program. VS was supported by the UNIQUE excellence scholarship (<https://www.unique.quebec/>) for a partial duration of the work. MJ is supported by a FRQNT Doctoral Fellowship (<https://doi.org/10.69777/366694>). SM acknowledges funding from FRQNT Doctoral Fellowship (<https://doi.org/10.69777/372208>).

This manuscript has been authored by Lawrence Livermore National Security, LLC under Contract No. DE-AC52-07NA27344 with the U.S. Department of Energy. This material is based upon work supported by the Department of Energy (LLNL-LDRD Program under Project No. 24-ERD-058, LLNL-CONF-2014473), Office of Science, Office of Advanced Scientific Computing Research. The United States Government retains, and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

References

- Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.
- Afra Amini, Tim Vieira, and Ryan Cotterell. Better estimation of the kullback–leibler divergence between language models. In *Neural Information Processing Systems*, 2025.
- Brian R. Bartoldson, Siddarth Venkatraman, James Diffenderfer, Moksh Jain, Tal Ben-Nun, Seanie Lee, Minsu Kim, Johan Obando-Ceron, Yoshua Bengio, and Bhavya Kailkhura. Trajectory balance with asynchrony: Decoupling exploration and learning for fast, scalable LLM post-training. In *Neural Information Processing Systems*, 2025.
- Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, et al. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*, 2024.
- Shiyi Cao, Sumanth Hegde, Dacheng Li, Tyler Griggs, Shu Liu, Eric Tang, Jiayi Pan, Xingyao Wang, Akshay Malik, Graham Neubig, Kourosh Hakhamaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-v0: Train real-world long-horizon agents via reinforcement learning, 2025.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *Transactions on Machine Learning Research*, 2026.
- Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, 2023.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shutong Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An,

- Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, 2025.
- Anthony GX-Chen, Jatin Prakash, Jeff Guo, Rob Fergus, and Rajesh Ranganath. KL-regularized reinforcement learning is designed to mode collapse. In *International Conference on Learning Representations*, 2026.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.
- Jian Hu, Xibin Wu, Weixun Wang, Xianyu, Dehao Zhang, and Yu Cao. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework. *arXiv preprint arXiv:2405.11143*, 2024.
- Prime Intellect. Prime-rl, 2025. URL <https://github.com/PrimeIntellect-ai/prime-rl>.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *International Conference on Machine Learning*, 2017.
- Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.
- Nathan Lambert. *Reinforcement Learning from Human Feedback*. Online, 2025. URL <https://rlhfbook.com>.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxu Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Conference on Language Modeling*, 2025.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *International Conference on Learning Representations*, 2023.
- 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. In *Conference on Language Modeling*, 2025.

- Michael Noukhovitch, Shengyi Huang, Sophie Xhonneux, Arian Hosseini, Rishabh Agarwal, and Aaron Courville. Faster, more efficient RLHF through off-policy asynchronous learning. In *International Conference on Learning Representations*, 2025.
- 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. *Neural Information Processing Systems*, 2022.
- Biqing Qi, Pengfei Li, Fangyuan Li, Junqi Gao, Kaiyan Zhang, and Bowen Zhou. Online dpo: Online direct preference optimization with fast-slow chasing. *arXiv preprint arXiv:2406.05534*, 2024.
- A Yang Qwen, Baosong Yang, B Zhang, B Hui, B Zheng, B Yu, Chengpeng Li, D Liu, F Huang, H Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.
- Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, 2014.
- Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII*, 2012.
- J. Schulman. Approximating kl divergence, 2020. URL <http://joschu.net/blog/kl-approx.html>.
- Vedant Shah, Dingli Yu, Kaifeng Lyu, Simon Park, Jiatong Yu, Yinghui He, Nan Rosemary Ke, Michael Mozer, Yoshua Bengio, Sanjeev Arora, et al. Ai-assisted generation of difficult math questions. *arXiv preprint arXiv:2407.21009*, 2024.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv: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. In *European Conference on Computer Systems*, 2025.
- 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. *Neural Information Processing Systems*, 2020.
- Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards. In *Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.
- Yunhao Tang and Rémi Munos. On a few pitfalls in kl divergence gradient estimation for rl. *arXiv preprint arXiv:2506.09477*, 2025.
- Emanuel Todorov. Linearly-solvable markov decision problems. *Neural Information Processing Systems*, 2006.
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. Left: Reasoning with reinforced fine-tuning. In *Annual Meeting of the Association for Computational Linguistics*, 2024.
- Jean Vassoyan, Nathanaël Beau, and Roman Plaud. Ignore the kl penalty! boosting exploration on critical tokens to enhance rl fine-tuning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, 2025.

- Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. Trl: Transformer reinforcement learning. <https://github.com/huggingface/trl>, 2020.
- Taiqiang Wu, Chaofan Tao, Jiahao Wang, Runming Yang, Zhe Zhao, and Ngai Wong. Rethinking kullback-leibler divergence in knowledge distillation for large language models. In *International Conference on Computational Linguistics*, pp. 5737–5755, 2025.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.
- Yifan Zhang, Yifeng Liu, Huizhuo Yuan, Yang Yuan, Quanquan Gu, and Andrew C Yao. On the design of kl-regularized policy gradient algorithms for llm reasoning. *arXiv preprint arXiv:2505.17508*, 2025.
- Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.
- Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

Appendix Contents

A	Related Work	16
B	Experimental Setup - Further Details	16
C	Reverse KL and Gradients for Parametric Autoregressive Model	17
D	Further Empirical Analysis	17
D.1	Additional Evaluation Results	17
D.2	Effect of Using Correct Gradient Estimators	18
D.3	Experiments on Code Reasoning	18
D.4	Further Results with Off-Policy Training	19
D.5	Further Analysis for $K1$ -in-reward and $K3$ -in-loss	19
E	$K1$ in Reward in Different Libraries	19
F	Mathematical Details and Derivations	20
F.1	True Gradient	20
F.2	Path-wise and Score Function Derivatives	20
F.3	$K1$ Estimator	21
F.4	$K3$ Estimator	22

A Related Work

KL divergence-based regularization in RL comes from the literature on KL control within stochastic optimal control (Todorov, 2006; Ziebart et al., 2008; Kappen et al., 2012; Rawlik et al., 2012). The KL regularization is typically applied as a penalty term in the reward function in KL control. In the context of sequence models, Jaques et al. (2017) first proposed using KL regularization to maintain information learned from data in pretraining, when training the model with RL with task-specific rewards. Ziegler et al. (2019) used this idea in the context of training language models with reward models learned from human preference data. In the context of reinforcement learning with learned reward models, KL regularization can also mitigate issues of reward-overoptimization (Gao et al., 2023), where the language model being trained can learn to exploit spurious correlations in the reward model and suffer catastrophic forgetting of the natural language abilities. KL regularization remains a key ingredient in the reinforcement learning from human feedback (RLHF) pipeline (Ouyang et al., 2022; Touvron et al., 2023).

For language models, computing the KL divergence is intractable. In practice, sample-based Monte-Carlo estimators are used to approximate the KL divergence to be added in the reward (Schulman, 2020). Early work on RLHF (Stiennon et al., 2020; Ouyang et al., 2022) for language models used the K1 estimator, whereas recently Shao et al. (2024) proposed using the K3 estimator, and Amini et al. (2025) proposed a Rao-Blackwellized estimator. Shao et al. (2024) also proposed incorporating the KL regularization term in the loss as opposed to the traditional location as part of the reward. Early work on RLHF (Ziegler et al., 2019; Stiennon et al., 2020) studied the effect of the strength of the KL regularization but there is no systematic study on the downstream effect of various estimators.

In the context of RL with verifiable rewards for LLM reasoning, the role of KL has received renewed interest. Guo et al. (2025) follow the GRPO recipe and propose using the K3 estimator in the loss. Some work, e.g., Yu et al. (2025) argue against using any KL regularization at all to allow the model to explore significantly and reduce memory usage. GX-Chen et al. (2026) re-examine the traditional intuition about the effects of forward and reverse KL in the context of LLM training and identify that the mode coverage instead depends primarily on factors such as regularization strength and relative scales between the rewards and the reference probabilities. Wu et al. (2025) also make the same observation in the case of on-policy distillation. Zhang et al. (2025) provide a unified formulation for regularized policy gradients for LLM training, and propose a new off-policy training objective. Vassoyan et al. (2025) propose ignoring KL penalties on critical tokens to boost exploration. Finally, Tang & Munos (2025) study some configurations of KL estimators used in RL training of LLMs and theoretically identify biases in the estimators. However, there is no systematic study covering all the estimators used in practice, analyzing the analytical forms of the estimators and gradients as well as the downstream impact on the model performance in the on and off-policy cases. Our study fills this gap.

B Experimental Setup - Further Details

For math experiments we use the training set of Hendrycks MATH (7500 questions) (Hendrycks et al., 2021) as the training set and its test set (5000 questions) as the test set for reporting training behaviors. For Countdown (Stojanovski et al., 2025) we generate train and held-out test sets consisting of 5000 examples each. We normalize the per-token RL objective by the number of tokens in the batch (Yu et al., 2025). We run the on-policy RL fine-tuning experiments on 2 GPU nodes consisting of 4 A100s (80GB) each. For the asynchronous RL fine-tuning experiments, we fine-tune Qwen3-4B-Instruct-2507 on 1 node with 4 80GB H100 GPUs, and Qwen2.5-7B on 2 nodes having 4 80GB H100 GPUs each. We use ver1 (Sheng et al., 2025) for the on-policy RL experiments and Prime-RL (Intellect, 2025) for the asynchronous RL experiments. For evaluation, we use lm-eval-harness (Biderman et al., 2024), using vLLM (Kwon et al., 2023) for inference with top_p = 1.0, temperature = 1.0 and min_p = 1.0.

C Reverse KL and Gradients for Parametric Autoregressive Model

The closed-form expressions for the reverse KL divergence and its gradient, corresponding to parametric autoregressive model in §3.3 can be written as

$$D_{\text{KL}}(A\|B) = \mathbb{E}_{Y \sim A} [\log A_{\theta}(Y) - \log B_{\phi}(Y)] \quad (14)$$

$$\frac{\partial}{\partial a} D_{\text{KL}}(A\|B) = \mathbb{E}_{Y \sim A} \left[\sum_{t=1}^T (y_t - p_t^A) (\log A_{\theta}(Y) - \log B_{\phi}(Y)) \right], \quad (15)$$

$$\frac{\partial}{\partial b} D_{\text{KL}}(A\|B) = \mathbb{E}_{Y \sim A} \left[\sum_{t=1}^T (y_t - p_t^A) c_{t-1} (\log A_{\theta}(Y) - \log B_{\phi}(Y)) \right]. \quad (16)$$

D Further Empirical Analysis

In this section, we provide additional results to further support the claims discussed in the main paper.

D.1 Additional Evaluation Results

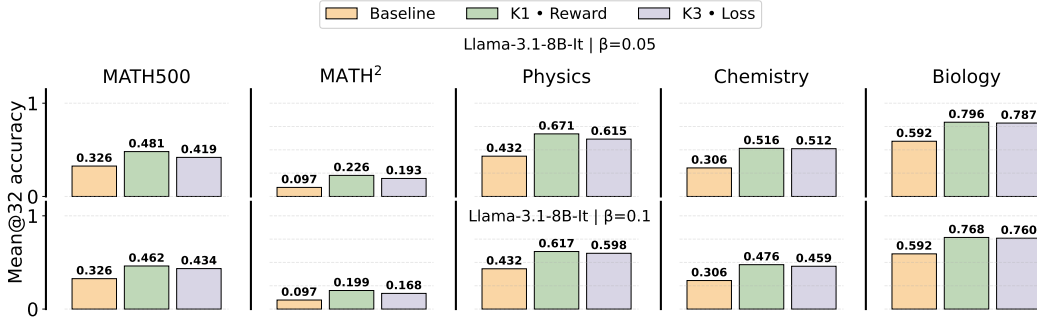


Figure 8: Comparison of Llama-3.1-8B-Instruct trained with two stable estimator configurations - **K1-in-reward** and **K3-in-loss**. Baseline refers to the performance of base Llama-3.1-8B-Instruct (prompted with chat template). **K1-in-reward** (unbiased gradient) performs the best on both in-domain and out-of-domain tasks. Increasing β deteriorates performance across the board.

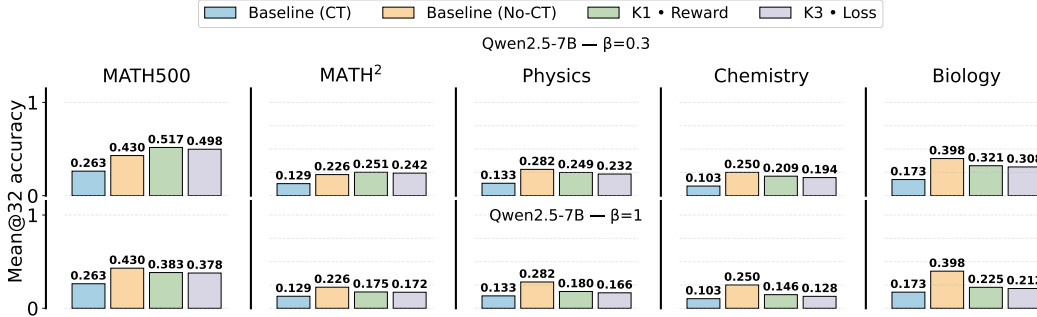


Figure 9: Comparison of Qwen2.5-7B trained with two stable estimator configurations - **K1-in-reward** and **K3-in-loss**. Baseline (CT) refers to the performance of base Qwen2.5-7B when prompted with a chat template. Baseline (No-CT) represents the performance when it is not prompted with a chat template. **K1-in-reward** (unbiased gradient) performs the best on both in-domain and out-of-domain tasks. Increasing β consistently deteriorates performance.

D.2 Effect of Using Correct Gradient Estimators

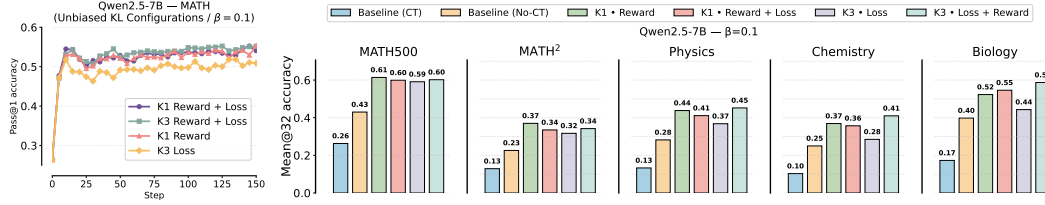


Figure 10: **Left: Train time performance of Qwen2.5-7B with different KL configurations ($\beta = 0.1$) on MATH test.** The configurations that give unbiased gradients perform similarly, and better than **K3-in-loss**. **Right: Performance of different KL configurations across several evaluation tasks.** Configurations resulting in unbiased gradients, (**K1-in-reward**, **K1-in-reward-and-loss** and **K3-in-reward-and-loss**), always outperform the biased configuration (**K3-in-loss**).

As noted in §2.2, adding the KL penalty in both reward as well as loss in on-policy settings would result in unbiased gradient estimates, regardless of the estimator. We test this empirically by training Qwen2.5-7B on MATH, completely on-policy ($\omega_t = 1$), using **K1** and **K3** added to both reward and loss, and $\beta = 0.1$ for 150 training steps with a learning rate of 10^{-6} . All other experimental details remain the same as in §4.1. We compare the performance of these models against other estimator configurations, which are trained for the same number of steps. Results are presented in Fig. 10. We observe that **K1-in-reward** performs the best on in-domain tasks, whereas **K3** added to both reward and loss performs the best on out-of-domain tasks. As a general observation, configurations inducing unbiased gradient estimates always outperform configurations inducing biased gradients, *i.e.*, **K3-in-loss**.

D.3 Experiments on Code Reasoning

We validate our observation in the case of on-policy training on a different setup. We train Qwen3-4B-Instruct-2507 on CodeIO (Stojanovski et al., 2025) Given a code snippet, the model is asked to predict the outputs given a set of inputs or vice-versa, without access to a sandbox. We use a batch size of 256, maximum response budget of 4096 tokens, 4 rollouts per prompt (group size), sampling temperature of 1.0, and a fixed KL coefficient of 0.1. The learning algorithm is RLOO. We train the model on a dataset of 15000 procedurally generated problems and evaluate the model on a held out test set. The test performance over the course of training is shown in Fig. 11. The observations in code reasoning remain consistent with the results in the main text: No KL performs best in this policy setting, while **K1** in loss and **K3** in reward tend to hurt performance.

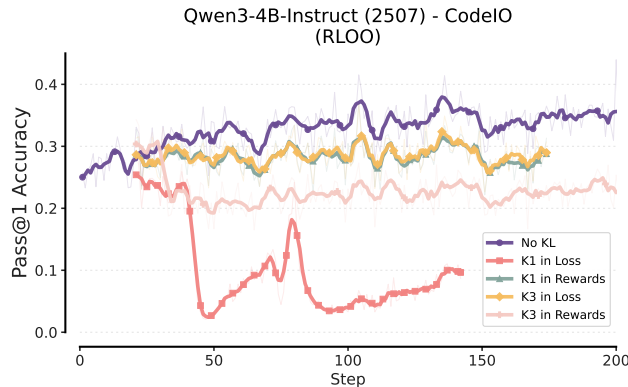


Figure 11: RL fine-tuning Qwen3-4B-Instruct-2507 on CodeIO with different KL configurations.

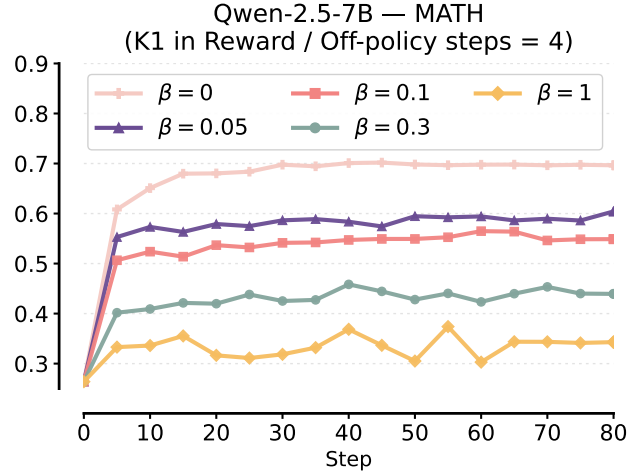


Figure 12: RL fine-tuning Qwen2.5-7B with **K1-in-reward**, total training batch size of 1024 and mini-batch size of 256. The training remains stable as opposed to configurations leading to biased gradients which result where the instabilities are accentuated as compared to on-policy training.

D.4 Further Results with Off-Policy Training

Figure 12 shows the performance of Qwen2.5-7B on MATH test set, while being fine-tuned on MATH training data, with a training batch size of 1024 and a mini-batch size of 256, i.e. $\omega \neq 1$, using **K1-in-reward**. Despite the off-policy, the training is stable as opposed to **K1-in-loss** and **K3-in-reward** with $\omega \neq 1$, where the training instabilities are accentuated as compared to the on-policy case.

D.5 Further Analysis for **K1-in-reward** and **K3-in-loss**

In an attempt to understand the performance difference between **K3-in-loss** and **K1-in-reward**, we plot two metrics: (1) the sequence-level forward KL divergence between the reference policy π_{ref} and the training policy π_{θ} , and (2) the entropy of the training policy π_{θ} . We compute these metrics for different checkpoints of Qwen2.5-7B while being trained on MATH, on one in-distribution task - MATH test dataset and one out-of-domain task - MMLU Biology. The results are shown in Figure 13. We observe that the forward KL divergence remains lower in the case of **K3-in-loss** whereas entropy stays lower for **K1-in-reward**. However, these observations do not directly explain the performance differences between **K1-in-reward** and **K3-in-loss** and further analysis needs to be carried out.

E K1 in Reward in Different Libraries

Table 2 lists the arguments to be included while submitting the training scripts to use the unbiased K1 in reward configuration discussed in this work for three popular libraries - VeRL (Sheng et al., 2025), OpenRLHF (Hu et al., 2024) and SkyRL (Cao et al., 2025). Another popular library under consideration - Prime-RL (Intellect, 2025) seems to have removed the previously used KL in loss term entirely.

Library	Arguments
VeRL	algorithm.use_kl_in_reward=True, algorithm.kl_penalty="kl"
OpenRLHF	DO NOT set --use_kl_loss, --kl_estimator="k1"
SkyRL	trainer.algorithm.kl_estimator_type="k1", trainer.algorithm.use_kl_in_reward=True

Table 2: KL regularization configuration across different libraries.

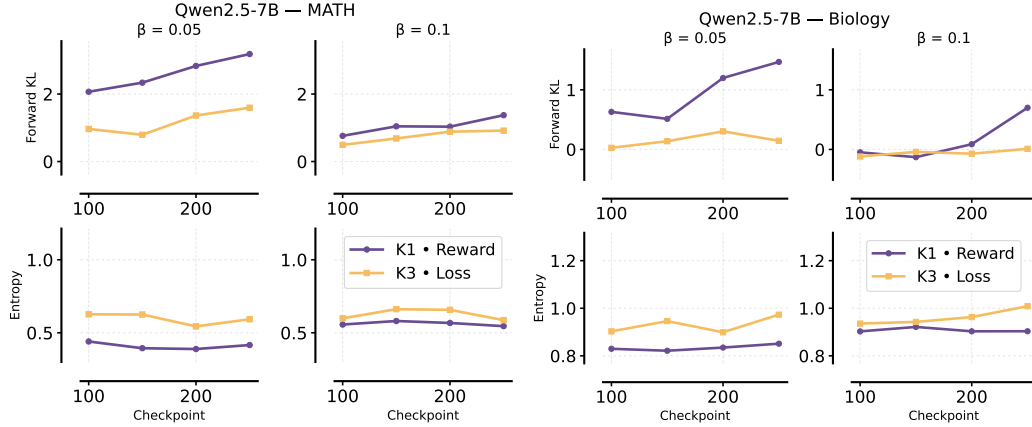


Figure 13: Sequence level forward KL with respect to the base policy and entropy across different training checkpoints, computed on MATH test dataset and MMLU Biology dataset

F Mathematical Details and Derivations

Notation. We first define the notation used for the analysis into the bias of the estimators and their corresponding gradients.

Symbol	Description
π_θ	policy trained using RL
π_{ref}	reference policy
x	prompt
$y_{1:T}$	generated response with T tokens
y_t	token at position t of response $y_{1:T}$
$K1$	naïve estimator of $D_{\text{KL}}(\pi_\theta \pi_{\text{ref}})$
$K3$	Schulman estimator of $D_{\text{KL}}(\pi_\theta \pi_{\text{ref}})$
$K1_t$	naïve estimator of $D_{\text{KL}}(\pi_\theta \pi_{\text{ref}})$ at token t
$K3_t$	Schulman estimator of $D_{\text{KL}}(\pi_\theta \pi_{\text{ref}})$ at token t

Table 3: Notation table.

F.1 True Gradient

We want to estimate the gradient of the KL divergence between π_θ and π_{ref} that are defined over entire sequences of tokens. Specifically, we want:

$$\nabla_\theta \mathbb{KL}(\pi_\theta(\cdot | x) || \pi_{\text{ref}}(\cdot | x)) = \nabla_\theta \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} \left[\log \frac{\pi_\theta(y_{1:T} | x)}{\pi_{\text{ref}}(y_{1:T} | x)} \right] \quad (17)$$

$$= \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} \left[\log \frac{\pi_\theta(y_{1:T} | x)}{\pi_{\text{ref}}(y_{1:T} | x)} \nabla_\theta \log \pi_\theta(y_{1:T} | x) \right]. \quad (18)$$

This is the true *sequence*-level gradient of the KL divergence. Every gradient estimator we use henceforth aims to estimate this true gradient.

F.2 Path-wise and Score Function Derivatives

We show how the gradient of the KL estimator, or any other function, decomposes into a *path-wise* derivative corresponding to the gradient of the estimator inside the expectation,

and the *score function* derivative arising from the θ -dependent sampling in the expectation.

$$\nabla_{\theta} \mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot|x)} [\widehat{\text{KL}}] \quad (19)$$

$$= \nabla_{\theta} \sum_{y_{1:T}} \widehat{\text{KL}} \cdot \pi_{\theta}(y_{1:T} | x) \quad (20)$$

$$= \sum_{y_{1:T}} \left(\nabla_{\theta} \widehat{\text{KL}} \right) \cdot \pi_{\theta}(y_{1:T} | x) + \sum_{y_{1:T}} \widehat{\text{KL}} \cdot (\nabla_{\theta} \pi_{\theta}(y_{1:T} | x)) \quad (21)$$

$$= \underbrace{\mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot|x)} \nabla_{\theta} [\widehat{\text{KL}}]}_{\text{path-wise derivative}} + \underbrace{\mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot|x)} [\widehat{\text{KL}} \cdot \nabla_{\theta} \log \pi_{\theta}(y_{1:T} | x)]}_{\text{score function derivative}} \quad (22)$$

$$= \underbrace{\mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot|x)} \left[\sum_t \nabla_{\theta} \widehat{\text{KL}}_t \right]}_{\text{Pathwise}} + \underbrace{\mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot|x)} \left[\left(\sum_t \widehat{\text{KL}}_t \right) \cdot \nabla_{\theta} \log \pi_{\theta}(y_{1:T} | x) \right]}_{\text{Score function}}, \quad (23)$$

where in the last line we write the KL divergence estimator as the sum of estimators at each individual token. Note that the path-wise derivative corresponds to using the estimator directly in the loss (and backpropagating through it), whereas the score function derivative corresponds to adding the estimator to the reward.

F.3 K1 Estimator

The **K1** estimator for a sequence $y_{1:T}$ can be written as:

$$\text{K1} = \sum_{t=1}^T \text{K1}_t = \sum_{t=1}^T \log \frac{\pi_{\theta}(y_t | x, y_{<t})}{\pi_{\text{ref}}(y_t | x, y_{<t})}. \quad (24)$$

It is easy to see that this is an unbiased estimator of $D_{\text{KL}}(\pi_{\theta} || \pi_{\text{ref}})$.

To analyze the gradient of **K1**, we calculate the path-wise derivative and the score function derivative separately.

Path-wise derivative. The path-wise derivative of **K1** evaluates to zero under expectation.

$$\mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot|x)} \left[\nabla_{\theta} \sum_t \text{K1}_t \right] = 0. \quad (25)$$

Score function derivative. The score function derivative of **K1** is an unbiased estimate of the true gradient in Eq. (18).

$$\begin{aligned} & \mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot|x)} \left[\left(\sum_t \text{K1}_t \right) \cdot \nabla_{\theta} \log \pi_{\theta}(y_{1:T} | x) \right] \\ &= \mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot|x)} \left[\log \frac{\pi_{\theta}(y_{1:T} | x)}{\pi_{\text{ref}}(y_{1:T} | x)} \nabla_{\theta} \log \pi_{\theta}(y_{1:T} | x) \right]. \end{aligned} \quad (26)$$

Therefore, adding **K1** estimator to the reward results in an unbiased estimate of the gradient of the KL-regularized RL objective, whereas using **K1** in the loss directly does not. In fact, since the path-wise derivative of **K1** is zero in expectation, in principle, using it in the loss should be equivalent to optimizing the RL objective without KL regularization (*i.e.*, $\beta = 0$). In practice, however, using this term in the loss introduces some variance that can hurt the optimization. We also note that we can reduce the variance of the score function derivative by removing the past tokens from the inner sum, since their contribution to the gradient will be zero in expectation.

Takeaway for K1:

- Adding **K1** to the reward gives us an unbiased estimate of the gradient of the KL-regularized RL objective.
- Using **K1** in loss results in a biased estimate of the true gradient and is equivalent to using no KL regularization, but can introduce some variance in practice.

F.4 K3 Estimator

The **K3** estimator for a sequence $y_{1:T}$ can be written as:

$$\mathbf{K3} = \sum_{t=1}^T \mathbf{K3}_t = \sum_{t=1}^T \left(\frac{\pi_{\text{ref}}(y_t \mid y_{<t}, x)}{\pi_{\theta}(y_t \mid y_{<t}, x)} - 1 - \log \frac{\pi_{\text{ref}}(y_t \mid y_{<t}, x)}{\pi_{\theta}(y_t \mid y_{<t}, x)} \right). \quad (27)$$

We first show that **K3** is an unbiased estimator of $D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}})$:

$$\mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot \mid x)} [\mathbf{K3}] \quad (28)$$

$$= \mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot \mid x)} \left[\sum_t \left(\frac{\pi_{\text{ref}}(y_t \mid x, y_{<t})}{\pi_{\theta}(y_t \mid x, y_{<t})} - 1 - \log \frac{\pi_{\text{ref}}(y_t \mid x, y_{<t})}{\pi_{\theta}(y_t \mid x, y_{<t})} \right) \right] \quad (29)$$

$$= \mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot \mid x)} \left[\sum_t \frac{\pi_{\text{ref}}(y_t \mid x, y_{<t})}{\pi_{\theta}(y_t \mid x, y_{<t})} \right] - T + \mathbb{E}_{y_{1:T} \sim \pi_{\theta}} \left[\log \frac{\pi_{\theta}(y_{1:T} \mid x)}{\pi_{\text{ref}}(y_{1:T} \mid x)} \right] \quad (30)$$

$$= D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\text{ref}}) \quad (31)$$

We again calculate the path-wise and the score function derivatives of **K3** separately.

Path-wise derivative. The path-wise derivative of **K3** is a biased estimate of the true gradient in Eq. (18).

$$\mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot \mid x)} [\nabla_{\theta} \mathbf{K3}_t] \quad (32)$$

$$= \mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot \mid x)} \left[\nabla_{\theta} \sum_t \left(\frac{\pi_{\text{ref}}(y_t \mid x, y_{<t})}{\pi_{\theta}(y_t \mid x, y_{<t})} - 1 - \log \frac{\pi_{\text{ref}}(y_t \mid x, y_{<t})}{\pi_{\theta}(y_t \mid x, y_{<t})} \right) \right] \quad (33)$$

$$= -\mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot \mid x)} \left[\sum_t \frac{\pi_{\text{ref}}(y_t \mid x, y_{<t})}{\pi_{\theta}(y_t \mid x, y_{<t})} \nabla_{\theta} \log \pi_{\theta}(y_t \mid x, y_{<t}) \right] + \mathbb{E}_{y_{1:T} \sim \pi_{\theta}(\cdot \mid x)} [\nabla_{\theta} \mathbf{K1}]$$

$$= -\sum_t \mathbb{E}_{y_{<t} \sim \pi_{\theta}(\cdot \mid x)} \mathbb{E}_{y_t \sim \pi_{\text{ref}}(\cdot \mid x, y_{<t})} [\nabla_{\theta} \log \pi_{\theta}(y_t \mid x, y_{<t})] \quad (34)$$

$$= \sum_t \mathbb{E}_{y_{<t} \sim \pi_{\theta}(\cdot \mid x)} [\nabla_{\theta} \text{KL}(\pi_{\text{ref}}(\cdot \mid x, y_{<t}) \parallel \pi_{\theta}(\cdot \mid x, y_{<t}))]. \quad (35)$$

The above expression resembles the gradient of the *forward* KL divergence at the token level, except that the samples are drawn from $\pi_{\theta}(\cdot \mid x)$ instead of $\pi_{\text{ref}}(\cdot \mid x)$.

Score function derivative. The score function derivative of [K3](#) also is a biased estimate of the true gradient in Eq. (18).

$$\begin{aligned} & \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} \left[\left(\sum_{t=1}^T \text{K3}_t \right) \cdot \nabla_\theta \log \pi_\theta(y_{1:T} | x) \right] \\ &= \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} \left[\left(\sum_t \left(\frac{\pi_{\text{ref}}(y_t | x, y_{<t})}{\pi_\theta(y_t | x, y_{<t})} - 1 - \log \frac{\pi_{\text{ref}}(y_t | x, y_{<t})}{\pi_\theta(y_t | x, y_{<t})} \right) \right) \nabla_\theta \log \pi_\theta(y_{1:T} | x) \right] \end{aligned} \quad (36)$$

$$\begin{aligned} &= \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} \left[\sum_t \frac{\pi_{\text{ref}}(y_t | x, y_{<t})}{\pi_\theta(y_t | x, y_{<t})} \nabla_\theta \log \pi_\theta(y_{1:T} | x) \right] \\ &\quad + \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} \left[\log \frac{\pi_\theta(y_{1:T} | x)}{\pi_{\text{ref}}(y_{1:T} | x)} \nabla_\theta \log \pi_\theta(y_{1:T} | x) \right] \end{aligned} \quad (37)$$

$$\begin{aligned} &= \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} \left[\sum_t \sum_s \frac{\pi_{\text{ref}}(y_t | x, y_{<t})}{\pi_\theta(y_t | x, y_{<t})} \nabla_\theta \log \pi_\theta(y_s | y_{<s}, x) \right] \\ &\quad + \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} \left[\log \frac{\pi_\theta(y_{1:T} | x)}{\pi_{\text{ref}}(y_{1:T} | x)} \nabla_\theta \log \pi_\theta(y_{1:T} | x) \right] \end{aligned} \quad (38)$$

$$= - \sum_t \mathbb{E}_{y_{<t} \sim \pi_\theta(\cdot | x)} [\nabla_\theta \text{KL}(\pi_{\text{ref}}(\cdot | x, y_{<t}) \| \pi_\theta(\cdot | x, y_{<t}))] + \mathbb{E}_{y_{1:T} \sim \pi_\theta(\cdot | x)} [\text{K1} \cdot \nabla_\theta \log \pi_\theta(y_{1:T} | x)] . \quad (39)$$

where in Equation 38 the terms corresponding to $s < t$ and $s > t$ reduce to 0. The first term in Eq. (39) represents the bias with respect to the true gradient. Therefore, using [K3](#) either in loss or added to the reward results in a biased estimate of the true gradient.

We note that the path-wise derivative of [K3](#) corresponds to the regularization term used in GRPO ([Shao et al., 2024](#)), a popular RL algorithm used for training LLMs.

Takeaway for [K3](#): Adding [K3](#) to the reward or using it in the loss results in a biased estimate of the gradient of the KL-regularized RL objective.