A COMEDY OF ESTIMATORS: ON KL REGULARIZATION IN RL TRAINING OF LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

The reasoning performance of large language models (LLMs) can be substantially improved by training them with reinforcement learning (RL). The RL objective for LLM training involves a regularization term, which is the reverse Kullback-Leibler (KL) divergence between the trained policy and the reference policy. Since computing the KL divergence exactly is intractable, various estimators are used in practice to estimate it from on-policy samples. Despite its wide adoption, including in several open-source libraries, there is no systematic study analyzing the numerous ways of incorporating KL estimators in the objective and their effect on the downstream performance of RL-trained models. Tang & Munos (2025) show that prevailing practices for incorporating KL regularization do not provide correct gradients for stated objectives, creating a discrepancy between the objective and its implementation. In this paper, we further analyze these practices and study the gradients of several estimators, revealing how design choices shape gradient bias. We substantiate these findings with empirical observations by RL fine-tuning `Qwen2.5-7B` and `Llama-3.1-8B-Instruct` with different configurations and evaluating their performance on both in- and out-of-distribution tasks. Through our analysis, we observe that: (1) estimator configurations with biased gradients can result in training instabilities; and (2) using estimator configurations resulting in unbiased gradients leads to better performance on in-domain as well as out-of-domain tasks. Overall, our findings provide useful takeaways for using KL-regularized objectives during RL post-training of LLMs.

1 INTRODUCTION

Reinforcement learning (RL) has become an indispensable component of present-day post-training pipelines for large language models (LLM). RL fine-tuning of LLMs was initially popularized for human preference alignment and instruction-following (Ouyang et al., 2022). Since then, RL has played a transformative role in reasoning-oriented post-training of LLMs. Recent work (Jaech et al., 2024; Guo et al., 2025) has shown that training LLMs as RL policies on reasoning tasks such as mathematics, coding, and open-ended reasoning leads to a substantial improvement in their performance. For this reason, there has been rapid progress in developing methods for reasoning-oriented training of LLMs using RL – the performance of fairly recently released reasoning models such as DeepSeek-R1 (Guo et al., 2025) is already being challenged by models with parameter counts lower by several orders (Yang et al., 2025). Much of this rapid progress has unfortunately been accompanied by inconsistent design decisions and implementation errors in RL fine-tuning pipelines (Tang & Munos, 2025).

One such design choice is the use of Kullback-Leibler (KL) divergence between the trained policy and the base policy as a regularization term in the objective (Peters et al., 2010; Ouyang et al., 2022). This regularization is crucial since it ensures that the policy explores within the space of coherent sequences by constraining it to the support of the base model, thus avoiding problems such as reward over-optimization (Gao et al., 2023) or catastrophic forgetting (McCloskey & Cohen, 1989; Qi et al., 2024) of the information present in the base model. Specifically, the *reverse* KL divergence is used for this regularization so that the policy assigns high probability mass to a narrow set of high-reward trajectories. This is opposed to the forward KL divergence, which tends to maintain probability mass over the entire support of the base model at the expense of performance. A regularization coefficient β controls the trade-off between reward maximization and proximity to the base model. However,

it is intractable to compute the reverse KL divergence exactly owing to the high-dimensionality of the space of possible sequences. As a result, different sample-based estimators of the reverse KL divergence are used in practice (Zhang et al., 2025; Amini et al., 2025).

In addition to differences in their approximations, these estimators may be incorporated into the objective in different ways: previous work doing 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 the KL term directly to the loss. The choice of the estimator, regularization coefficient, and whether it is added to the reward or directly to the loss has a significant effect on the training stability, convergence rate, and out-of-distribution generalization of the trained models. Moreover, recent work (Tang & Munos, 2025) identified that some of these practices 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. These issues have propagated to widely used public libraries, leading to potentially incorrect results when using KL regularization (Sheng et al., 2025; von Werra et al., 2020; Hu et al., 2024; Cui et al., 2025). These findings highlight that while KL regularization is ubiquitous in RL training of LLMs, the implementation details are poorly understood and often overlooked.

In this work, we attempt to fill this gap by providing a systematic exploration of the space of some design choices associated with the practical use of KL regularization. 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 – the naïve or `K1` estimator, and the Schulman or the `K3` estimator (Schulman, 2020). First, we analytically study the bias of the gradients with respect to the true gradient when these estimates are added to the reward versus when directly added to the loss (§3, Table 1). Next, we empirically investigate the bias of gradient estimates in each case in a synthetic setting (§4.1), substantiating our prior discussion. Finally, we perform experiments to study how these choices affect RL based fine-tuning of `Qwen2.5-7B` (Yang et al., 2024) and `Llama-3.1-8B-Instruct` (Touvron et al., 2023) on a mathematical reasoning task across different values of the KL regularization coefficient β, and study both in- and out-of-domain performance of the resulting models (§4.2).

Key observations:
- Unbiased estimates of the reverse KL divergence can result in *biased* gradients depending on their usage.
- Configurations inducing biased gradients often lead to unstable training and can precipitate complete collapse.
- Configurations that lead to unbiased gradient estimates result in better-performing models, across both in-domain and out-of-domain evaluation tasks.

2 Background

We study the problem of fine-tuning a base language model π_{ref} 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

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

where β is a hyperparameter that controls the weight of the KL divergence penalty, π_θ is the RL policy initialized at π_{ref} and $y_{1:T}$ denotes solutions generated by the model conditioned on the question, *i.e.* $y_{1:T} \sim \pi_\theta(\cdot|x)$. Since both the sampling of $y_{1:T}$ and the definition of 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).

The reward function R can be a learned model trained on human feedback data (Ouyang et al., 2022) or can be an oracle verifier for tasks such as math, games, and code. We focus on the latter setting, commonly termed Reinforcement Learning with Verifiable Rewards (RLVR). An example of a verifiable reward function for math is a comparison against the ground truth answer $R(y_{1:T}, y) = \mathbb{1}_{\text{Extract}(y_{1:T})=y}$, where Extract$(\cdot)$ pulls the solution from a `\boxed{}` format. In general this could be any reward model with y specifying the meta-data required for reward computation.

The objective in (1) incentivizes the policy π_θ to maximize the expected reward while remaining close to the base model π_{ref}. Within the context of LLMs, π_θ is constrained to remain close to the reference model π_{ref} to avoid over-optimization and catastrophic forgetting of the general capabilities of the base policy (Gao et al., 2023). The constraint is enforced via a *reverse* KL divergence penalty in the objective which is an expectation under the learned policy π_θ:

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

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

$$\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}\mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[A(x, y, y_{1:T})\right] - \beta D_{\text{KL}}(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x)) \tag{3}$$

Group Region Policy Optimization (GRPO) (Shao et al., 2024; Guo et al., 2025) is the most widely used algorithm for training π_θ in the context of RLVR. To compute the advantage $\hat{A}$, GRPO uses a group of samples $\{o_1, o_2, \cdots, o_G\}$ for each prompt $x \in D$. Like proximal policy optimization (PPO; Schulman et al., 2017), GRPO generates a set of samples from the policy and makes updates over minibatches, introducing a delay between the sampling policy ($\pi_{\theta_{\text{old}}}$) and π_θ.

$$\begin{aligned}\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\Bigg\{\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\Big[\frac{\pi_\theta(o_{i,t} \mid x, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid x, o_{i,<t})}\hat{A}_{i,t},\\ \text{clip}\left(\frac{\pi_\theta(o_{i,t} \mid x, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid x, o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\Big] - \beta D_{\text{KL}}\left(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x)\right)\Bigg\}\Bigg]\end{aligned} \tag{4}$$

where G is the total number of sequences sampled per *group*, and ε is a constant hyperparameter controlling the trust region for policy updates (Schulman et al., 2017). Notably, GRPO includes the KL term into the loss function instead of adding it to the reward. In order to restrict our study on the effect of the KL estimators, we opt to use REINFORCE leave-one-out (Ahmadian et al., 2024, RLOO) in our experiments. The only effective difference between GRPO and RLOO disregarding KL is the lack of both advantage and sequence-length normalization in the latter.

3 Overview of KL estimators and their gradients

In this work, we operate at the token level for the RL objective (Yu et al., 2025), as used in popular public libraries (Sheng et al., 2025). We are interested in estimating the reverse KL divergence between the *sequence-level distributions* $\pi_\theta(y_{1:T} \mid x)$ and $\pi_{\text{ref}}(y_{1:T} \mid x)$. It is commonly estimated by decomposing into token-level estimates, which can be seen as a Rao-Blackwellized estimator of the sequence-level reverse KL divergence, and has been shown to reduce variance (Amini et al., 2025). For a generic sequence-level reverse KL divergence estimator $\widehat{\text{KL}}$, we can write:

$$D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}) = \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\widehat{\text{KL}}\right] = \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\sum_{t=1}^{T}\widehat{\text{KL}}_t\right], \tag{5}$$

where $\widehat{\text{KL}}_t$ is the estimator defined on the *token-level distributions* $\pi_\theta(y_t \mid x, y_{<t})$ and $\pi_{\text{ref}}(y_t \mid x, y_{<t})$. 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:

$$\nabla_\theta\mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\widehat{\text{KL}}\right] = \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\sum_{t=1}^{T}\nabla_\theta\widehat{\text{KL}}_t + \sum_{t=1}^{T}\widehat{\text{KL}}_t\nabla_\theta\log\pi_\theta(y_{1:T} \mid x)\right]. \tag{6}$$

Such gradient estimators have been previously studied in (Ranganath et al., 2014; Tang & Munos, 2025). We refer the reader to Appendix E.2 for a derivation of (6). Next, we discuss two ways of using the estimator in the context of RL training of LLMs. Henceforth, we assume that we always sample on-policy, *i.e.*, $\omega = 1$.

[1]The 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.** 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. $r = \frac{\pi_{\text{ref}}(y_t|x,y_{<t})}{\pi_\theta(y_t|x,y_{<t})}$ in the expressions given.

Estimator	Expression	Position	Unbiased Grad. Est. (§3)	Behavior (§4.2)
K1	$-\log r$	Reward	✔	Stable
		Loss	✗	Training Instabilities
K3	$r - 1 - \log r$	Reward	✗	Training collapse
		Loss	✗	Stable

3.1 Position of the 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 advantage is then computed as follows:

$$r_t = s_t - \beta \text{sg}\left[\widehat{\text{KL}}_t\right], \qquad A_t = \sum_{t=1}^{T} r_t - b = R - \beta \sum_{t=1}^{T} \text{sg}\left[\widehat{\text{KL}}_t\right] - b, \tag{7}$$

where s_t is the token-level task score, usually 0 for intermediate tokens and either 1 or 0 for the final token depending on whether the sequence led to the correct answer, A_t is the advantage assigned at token t, $R = \sum_{t=1}^{T} s_t$, and b is the advantage baseline. The gradient of the objective $J(\theta)$ is:

$$\nabla_\theta J(\theta) = \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[(R - \beta \sum_{t=1}^{T} \widehat{\text{KL}}_t - b)\nabla_\theta \log \pi_\theta(y_{1:T} \mid x)\right]. \tag{8}$$

Loss. This refers to adding the KL estimator directly to the loss, 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. Thus, the gradient of the objective is computed as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[(R - b)\nabla_\theta \log \pi_\theta(y_{1:T} \mid x) - \beta \sum_{t=1}^{T} \nabla_\theta \widehat{\text{KL}}_t\right] \tag{9}$$

Note that the gradient contribution of the KL estimator when used in the reward is $\sum_{t=1}^{t=T} \widehat{\text{KL}}_t \nabla_\theta \log \pi_\theta(y_{1:T} \mid x)$ and when used in the loss is $\sum_{t=1}^{T} \nabla_\theta \widehat{\text{KL}}_t$, scaled by β in both cases. Therefore, in the general case, both of these terms in isolation are biased with respect to the correct gradient as stated in (6). However, we can always recover the correct gradient by adding the estimator to both the reward and the loss.

3.2 Inspecting KL estimators

We start with the knowledge that the true gradient of the reverse KL divergence is:

$$\nabla_\theta \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\widehat{\text{KL}}\right] = \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\log \frac{\pi_\theta(y_{1:T} \mid x)}{\pi_{\text{ref}}(y_{1:T} \mid x)} \nabla_\theta \log \pi_\theta(y_{1:T} \mid x)\right]. \tag{10}$$

We now examine two estimators commonly used in RL training of LLMs, namely the naïve or K1 estimator, and the Schulman or K3 estimator (Schulman, 2020). For each of these estimators, we derive the gradient of its expectation when used in reward and in loss, and determine their bias by comparing them against the true gradient in (10).

3.2.1 K1 estimator

The K1 estimator is computed as the Monte Carlo estimate of the log-ratio of likelihoods under the current and reference policies, with samples from the current training policy. We can write K1 as the

sum of token-level log ratios, which we denote by $\mathtt{K1}_t$:

$$\mathtt{K1} = \sum_{t=1}^{T} \mathtt{K1}_t = \sum_{t=1}^{T} \log \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\text{ref}}(y_t \mid x, y_{<t})} \tag{11}$$

We now analyze the gradients resulting from using K1 estimator, both in the case of adding to the reward (Equation (8)) and the loss (Equation (9)). We refer the reader to Appendix E.3 for a derivation of the gradients in two configurations.

Reward. Note that $\sum_{t=1}^{T} \mathtt{K1}_t = \log \frac{\pi_\theta(y_{1:T}|x)}{\pi_{\text{ref}}(y_{1:T}|x)}$. The expected gradient (under π_θ) of the K1 estimator when used in the reward is unbiased with respect to the reverse KL gradient. The gradient of K1-in-reward is shown below.

$$\nabla_\theta \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\sum_{t=1}^{T} \mathtt{K1}_t \right] = \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\log \frac{\pi_\theta(y_{1:T} \mid x)}{\pi_{\text{ref}}(y_{1:T} \mid x)} \nabla_\theta \log \pi_\theta(y_{1:T} \mid x) \right] \tag{12}$$

Loss. Adding K1 to the loss results in the gradient being zero in expectation, and therefore, is biased.

$$\nabla_\theta \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\sum_{t=1}^{T} \mathtt{K1}_t \right] = \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\nabla_\theta \log \frac{\pi_\theta(y_{1:T} \mid x)}{\pi_{\text{ref}}(y_{1:T} \mid x)} \right] = 0 \tag{13}$$

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. This is because the first term of equation Equation (6) is zero in expectation for the K1 estimator as shown in (13).

3.2.2 K3 ESTIMATOR

The K3 estimator, similar to the K1 estimator, is unbiased (see Appendix E.4 for a proof). However, it also has a lower variance and thus is often preferred over K1 in practice. We can write K3 as:

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

Below, we state the gradient of the expectation of the estimator when K3 is used in reward and loss. We refer the reader to Appendix E.4 for a derivation of the gradients in two configurations for K3.

Reward. The gradient of the expectation of the KL estimate when K3 is used in the reward is:

$$\nabla_\theta \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\sum_{t=1}^{T} \mathtt{K3}_t \right] = \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\sum_{t=1}^{T} \left(\frac{\pi_{\text{ref}}(y_t|x,y_{<t})}{\pi_\theta(y_t|x,y_{<t})} + \log \frac{\pi_\theta(y_{1:T}|x)}{\pi_{\text{ref}}(y_{1:T}|x)} \right) \nabla_\theta \log \pi_\theta(y_{1:T} \mid x) \right] \tag{14}$$

Clearly, it is biased by the term $\mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\sum_{t=1}^{T} \frac{\pi_{\text{ref}}(y_t|x,y_{<t})}{\pi_\theta(y_t|x,y_{<t})} \nabla_\theta \log \pi_\theta(y_{1:T} \mid x) \right]$.

Loss. The gradient of the expectation of the KL estimate when K3 is used in the loss is as given below. This is the version used in the implementations of some of the most popular RL algorithms, such as GRPO (Shao et al., 2024; Guo et al., 2025). It is a biased estimate of the true reverse KL gradient shown in (6).

$$\nabla_\theta \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\sum_{t=1}^{T} \mathtt{K3}_t \right] = \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\sum_{t=1}^{T} \left(-\frac{\pi_{\text{ref}}(y_t \mid x, y_{<t})}{\pi_\theta(y_t \mid x, y_{<t})} \right) \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t}) \right] \tag{15}$$

Table 1 summarizes the two estimators and the biasedness of their gradients in different settings.

4 EMPIRICAL OBSERVATIONS

In this section, we complement our analysis in §3 with an empirical study on the effect of various configurations of KL estimators. In §4.1, we analyze the bias and variance of different configurations with a simple parametric autoregressive model (reinforcing the discussion in §3). Next, in §4.2, we study the effect of various configurations of KL estimators for RL fine-tuning of Qwen2.5-7B and Llama-3.1-8B models.

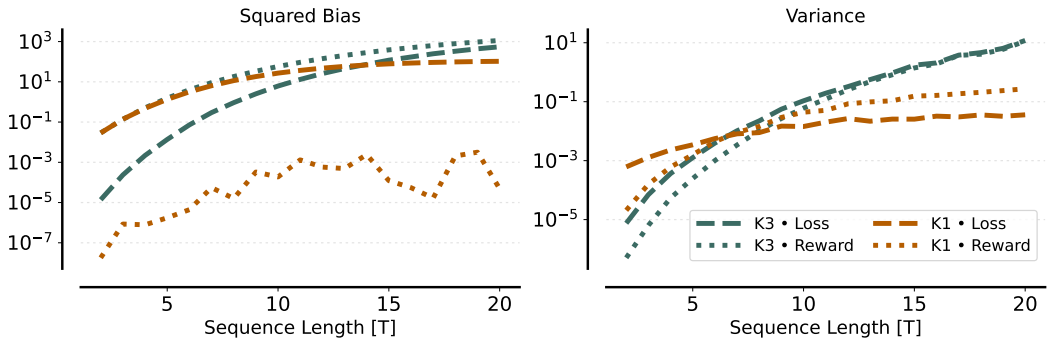

Figure 1: **The bias and variance of expected gradients with respect to the parameters of A, in different configurations (logarithmic scale).** While all estimators are unbiased, the expected gradients are unbiased only in the case of `K1` estimator when used in reward. `K3` estimator when used in reward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

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

$$A_\theta(Y) = \prod_{t=1}^{T} (p_t^A)^{y_t} (1 - p_t^A)^{1-y_t}, p_t^A = \sigma\big(a + b\, c_{t-1}\big), c_{t-1} = \sum_{k=1}^{t-1} y_k \tag{16}$$

$$B_\phi(Y) = \prod_{t=1}^{T} (p_t^B)^{y_t} (1 - p_t^B)^{1-y_t}, p_t^B = \sigma\big(\tilde{a} + \tilde{b}\, c_{t-1}\big), c_{t-1} = \sum_{k=1}^{t-1} y_k \tag{17}$$

where 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 $D_{\mathrm{KL}}(A \,\|\, B)$ and its gradient with respect to a and b admit closed-form expressions (see appendix C).

We compute the KL divergence with different estimators and their gradients when used as reward and in the loss as discussed in §3, 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 §3.

4.2 RL fine-tuning of LLMs

We now substantiate our takeaways from §3 and §4.1 through empirical analysis on RL fine-tuning of Qwen2.5-7B and Llama-3.1-8B-Instruct models with various KL in all configurations.

Experimental Setup. We RL fine-tune models on the training subset of Hendrycks MATH (Hendrycks et al., 2021) (henceforth referred to as MATH) consisting of 7500 problems. We use a token-level implementation (Yu et al., 2025) of REINFORCE with a leave one out advantage baseline (Ahmadian et al., 2024) as the policy gradient objective, with the various KL estimator configurations as discussed above. During training, we set both total training batch size and mini-batch size to be equal to 256 (*i.e.*number of policy update steps per sampled batch = 1), unless stated otherwise, avoiding any off-policy updates. We evaluate the models on 2 different in-domain tasks, namely MATH500 (Lightman et al., 2023) (500 examples) and MATH^2 (Shah et al., 2024) (210 examples) and 3 out-of- domain tasks – MMLU college physics (118 examples), college chemistry (113 examples) and college biology (165 rows) subsets (Hendrycks et al., 2020). The tasks are selected to analyze the effect of different estimator configurations on reasoning as well as non-reasoning (*i.e.*knowledge recall-oriented) tasks. We report the Pass@1 score of the model on the complete MATH test set (5000 examples) for training progress, and report mean@32 accuracy

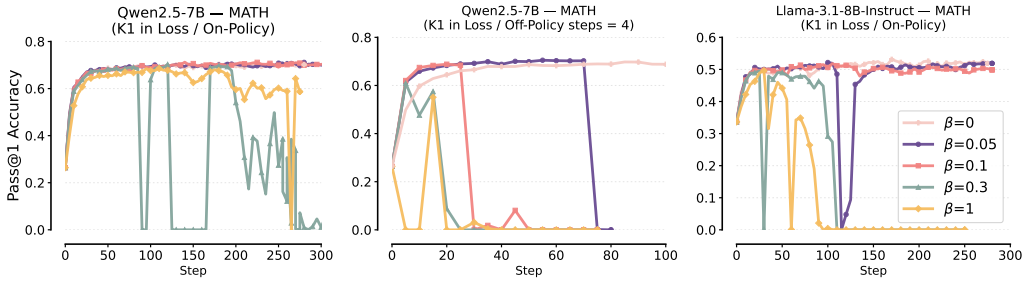

Figure 2: **Training Instabilities when using K1 in loss.** Pass@1 performance for **[Left]** training Qwen2.5-7B with $\texttt{K1}_t$ leads to training instabilities for $\beta = 0.1$ and 1. **[Center]** Training Qwen2.5-7B with 4 policy update steps per sampled batch accentuates the instabilities owing to the increased off-policyness, leading to definitive training collapse in all cases. **[Right]** Training Llama-3.1-8B-Instruct with $\texttt{K1}_t$ in loss leads to instabilities for all β except 0.1.

across 3 seeds to report evaluation performance of the models. We use default chat-templates while fine-tuning and evaluating models. Qwen2.5-7B (non-RL fine-tuned) is evaluated both with and without the chat template. More details about the experimental setup are discussed in Appendix B.

Observation 1: *Adding* K1 *estimator to the loss leads to training instabilities.*

As shown in (13), adding $\texttt{K1}_t$ 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 $\texttt{K1}_t$ 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 (Yang et al., 2024) and Llama-3.1-8B (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.

We observe training instabilities with $\beta = 0.3$ and 1 for Qwen models (Fig. 2 (left)) and all β except 0.1 for Llama models (Fig. 2 (right)). A potential explanation is that the term $\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})$, despite having an expectation of 0, adds variance to the optimization, leading to instabilities. Additionally, we observe that moving away from the default setting of fully on-policy updates, to off-policy updates (4 minibatch updates over each sampled batch) (Fig. 2 (center)) accentuates the instabilities and leads to consistent training collapse across all β even for Qwen modelsFurther, 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 models seem to be more robust to variance as compared to Llama-3.1-Models.

Observation 2: *Adding* K3 *estimator to the reward leads to training collapse.*

Another case of biased gradient estimate is K3 used in the reward. From eq. (14) we observe that adding token-level K3 to rewards leads to a bias term of

$$\mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\sum_{t=1}^{T} \frac{\pi_{\text{ref}}(y_t \mid x, y_{<t})}{\pi_\theta(y_t \mid x, y_{<t})} \nabla_\theta \log \pi_\theta(y_{1:T} \mid x)\right] \tag{18}$$

This bias is illustrated in Fig. 1 for the parametric autoregressive model §4.1. To validate this with LLMs, we RL fine-tune Qwen2.5-7B and Llama-3.1-8B-Instruct models with $\beta = 0.05, 0.1, 0.3$ and 1. Fig. 3 shows that this biased gradient estimate leads to unpredictable behavior leading to complete or partial collapse of the training for all β.

Observation 3: *Unbiased gradient estimators lead to 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 reverse KL is when K3 is used in the loss eq. (15). Surprisingly despite the bias, using K3 in the loss exhibits stable training of the policy. Fig. 4 (Left) 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 (15) in this case is a sum of unbiased gradient estimate of the forward KL divergences computed at the token level, making this configuration equivalent to a stable forward KL-based logit

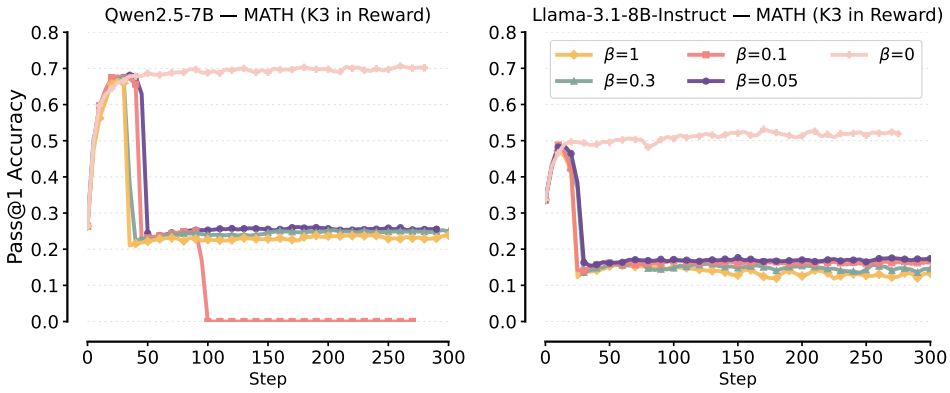

Figure 3: **Collapse in the case of adding K3 to the reward.** Pass@1 performance for **[Left]** Qwen2.5-7B trained on MATH train dataset **[Right]** Llama-3.1-8B-Instruct trained on MATH train dataset. The collapse maybe attributed to high bias and variance of the configuration.

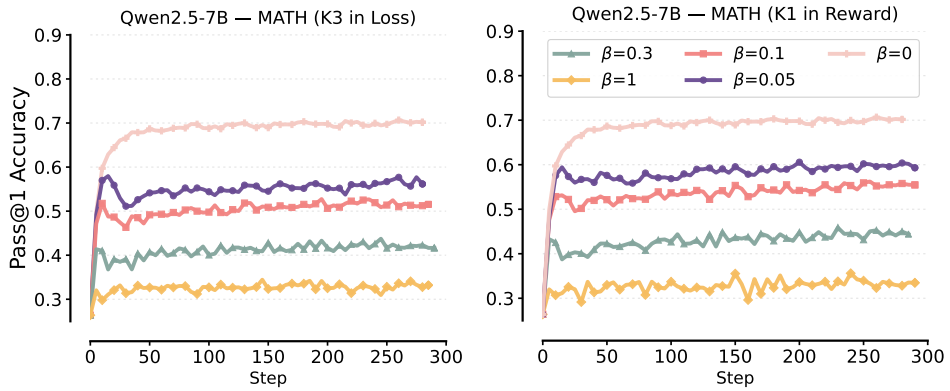

Figure 4: Pass@1 performance on **MATH test set with K3-in-loss (biased; Left) and K1-in-reward (unbiased; Right).** Although biased with respect to reverse KL, K3-in-loss yields stable training. In both cases, lower β values lead to higher performance.

distillation objective trained on-policy, with the base model as a teacher. 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). Comparing to Fig. 4 (Right), the in-distribution performance in this biased case is similar to the performance when training with the unbiased gradient estimator setting of adding K1 to the reward. Note that while $\beta = 0$ seems to work the best in this experimental setting, it may not always be the case, even within the paradigm of RLVR. We discuss one such example in appendix D.1.

Further, we compare the downstream performance of the models trained with K3 in loss and K1 in reward. We train Qwen2.5-7B and Llama-3.1-8B-Instruct models with different estimator configurations for 250 steps on MATH train data, and compare the performance of different configurations on a wide range of evaluation tasks as shown in Fig. 5 and Fig. 6 (similar results for $\beta = 0.3$ and $\beta = 1$ can be found in appendix D.2. Apart from MATH500 which is in-training distribution, we evaluate on MATH^2 which is out-of-distribution (OOD) but in-domain, as well as MMLU subsets of college-physics, college-biology and college-chemistry, all OOD tasks.

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 Qwen-2.5-7B, 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 across MATH500 and MATH^2, 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%). Similar trends hold for $\beta = 0.1$ as well. Consistent performance improvements iin the case of unbiased estimated gradient implementations over biased estimated

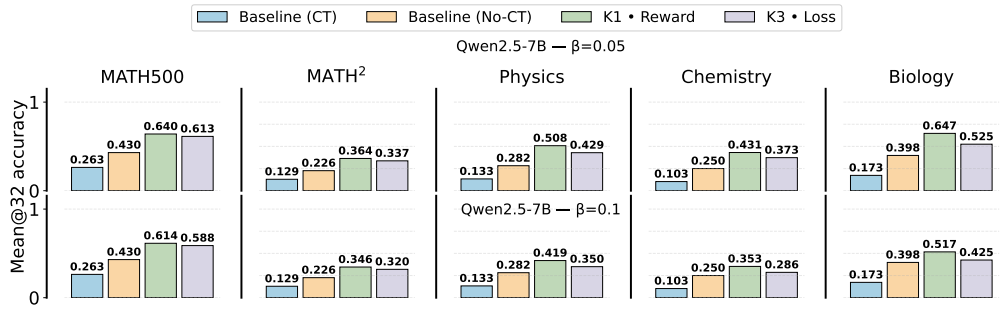

Figure 5: **Comparison of Qwen2.5-7B trained with two stable estimator configurations - K1 in reward and K3 in loss.** Baseline (CT) refers to the performance of base Qwen2.5-7B when prompted with a chat template. Baseline (No-CT) represents the performance when it is prompted with a chat template. K1 in Loss (unbiased gradient performs the beats on both in-domain and out-of-domain tasks. Increasing β consistently deteriorates performance.

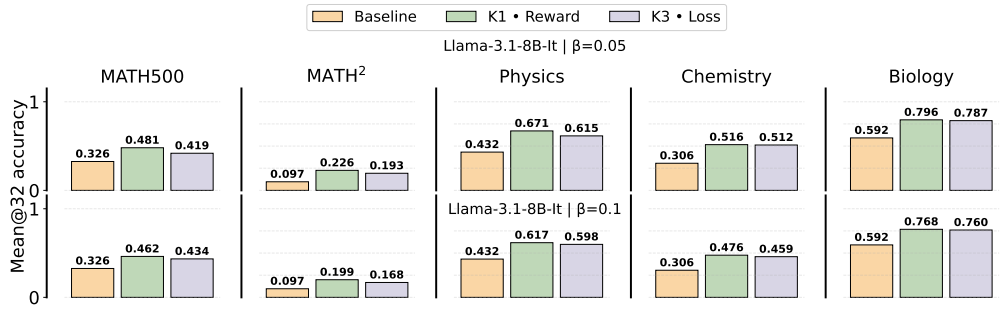

Figure 6: **Comparison of Llama-3.1-8B-Instruct trained with two stable estimator configurations - K1 in reward and K3 in loss.** Baseline refers to the performance of base Llama-3.1-8B-Instruct (prompted with chat template). K1 in Loss (unbiased gradient performs the beats on both in-domain and out-of-domain tasks. Increasing β deteriorates performance across the board.

gradient implementations demonstrate the importance of using correct gradient estimates while incorporating KL-based regularization.

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.

These findings should hardly be surprising: updating parameters with a step that is not the gradient of the desired objective – or indeed of any objective, if it is not a gradient field – is a recipe for instability, as stable behavior near an optimum is not guaranteed. However, the prevalence of these incorrect estimators in the literature and in implementations, and the fact that they *sometimes* work well enough to be published and overlooked in popular libraries, suggests that there is a lack of awareness of these issues. We hope that our systematic study will help clarify these issues for the community.

Future work: Much interesting analysis remains to be done in understanding the reasons for training instabilities, as well as studying the effect of estimators in settings with non-verifiable rewards (learned proxy rewards) which are more susceptible to reward hacking during RL fine-tuning (Gao et al., 2023). Further, the biased configurations discussed in this work could be made unbiased by including appropriate correction terms in the objective. Future work can study the behavior of these corrected implementations with the unbiased configuration studied in this paper.

REPRODUCIBILITY STATEMENT

We provide all the details to reproduce our results in §4.2 and appendix B.

LLM USE

LLMs were used to assist in writing code for experiments in the paper. No LLMs were used to assist with writing and formatting of the paper.

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. *arXiv preprint arXiv:2402.14740*, 2024.

Afra Amini, Tim Vieira, and Ryan Cotterell. Better estimation of the kl divergence between language models. *arXiv preprint arXiv:2504.10637*, 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.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025. URL https://arxiv.org/abs/2502.01456.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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, Shuting 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.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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, 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. doi: 10.48550/arXiv.2405.11143. URL https://arxiv.org/abs/2405.11143.

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.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Unlocking RL potential for LLM reasoning through refined credit assignment, 2025. URL https://openreview.net/forum?id=5mJrGtXVwz.

Tomasz Korbak, Ethan Perez, and Christopher L Buckley. Rl with kl penalties is better viewed as bayesian inference. *arXiv preprint arXiv:2205.11275*, 2022.

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, Xinxi 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 *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=i1uGbfHHpH.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. *arXiv preprint arXiv:2310.10505*, 2023.

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 *The Twelfth International Conference on Learning Representations*, 2023.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

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.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1607–1612, 2010.

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.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial intelligence and statistics*, pp. 814–822. PMLR, 2014.

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

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

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 *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 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. *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.

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. Reft: Reasoning with reinforced fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7601–7614, 2024.

Jean Vassoyan, Nathanaël Beau, and Roman Plaud. Ignore the kl penalty! boosting exploration on critical tokens to enhance rl fine-tuning. *arXiv preprint arXiv:2502.06533*, 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.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Taiqiang Wu, Chaofan Tao, Jiahao Wang, Runming Yang, Zhe Zhao, and Ngai Wong. Rethinking kullback-leibler divergence in knowledge distillation for large language models. *arXiv preprint arXiv:2404.02657*, 2024.

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.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yi-Chao Zhang, Yunyang Wan, Yuqi Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, Shanghaoran Quan, and Zekun Wang. Qwen2.5 technical report. *ArXiv*, abs/2412.15115, 2024. URL https://api.semanticscholar.org/CorpusID:274859421.

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

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

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

A Related work

Over the past few years, researchers have explored a variety of strategies to strengthen the reasoning capabilities of LLMs. Broadly, these strategies fall into three categories: *pre-training*, which equips models with general reasoning ability through large-scale unsupervised learning (Kaplan et al., 2020); *fine-tuning*, which adapts models on curated reasoning-oriented datasets (Hendrycks et al., 2020; Wei et al., 2022; Shao et al., 2024; Grattafiori et al., 2024; Touvron et al., 2023); and *prompting*, which improves reasoning through carefully designed input strategies without altering model parameters (Wei et al., 2022; Lightman et al., 2023). We focus on fine-tuning methods, and in particular investigate how KL-based interventions affect reasoning performance across models and datasets.

While fine-tuning can improve task-specific reasoning, a central challenge is *catastrophic forgetting*; models may lose general abilities acquired during pre-training when optimized on narrow domains (Ouyang et al., 2022). Aggressive fine-tuning on small or biased datasets can also cause overfitting or undesirable behaviors. To address these risks, researchers employ regularization methods (Korbak et al., 2022; Peters et al., 2010; Schulman, 2020). Common practices include using smaller learning rates, freezing subsets of parameters, or mixing in pre-training data during fine-tuning (Touvron et al., 2023; Grattafiori et al., 2024; Penedo et al., 2024).

A particularly effective regularization technique is the use of *Kullback–Leibler (KL) divergence* penalties. KL regularization is widely used in reinforcement learning from human feedback (RLHF), where it serves as a safety mechanism to prevent the fine-tuned model from drifting too far from the base model (Christiano et al., 2017; Ziegler et al., 2019; Ouyang et al., 2022; Stiennon et al., 2020; Lambert, 2025). In RLHF, the fine-tuned model (policy) is optimized to maximize a reward model score *minus* a KL penalty that measures divergence from the base LM distribution. This prevents reward hacking and ensures outputs remain fluent and human-like.

RL with KL control is used explicitly to strengthen *reasoning*. In mathematical reasoning, Guo et al. (2025); Shao et al. (2024) introduce GRPO (a critic-free PPO variant) and *add the KL term directly to the loss*, reporting substantial gains on GSM8K and MATH. Sequence-level objectives that preserve the KL-shaped reward also appear competitive for preference-tuned reasoning models, with RLOO showing robustness across tasks and reduced sensitivity to KL settings compared to PPO (Ahmadian et al., 2024; Li et al., 2023; Zheng et al., 2025; Yu et al., 2025; Kazemnejad et al., 2025). .

Although the role of KL regularization is widely acknowledged as important for reasoning fine-tuning, few works have systematically explored it in depth. Recent theoretical analyses underscore both the promise and the limitations of KL-based interventions. Amini et al. (2025) propose improved estimation techniques for KL between LLMs, Zhang et al. (2025) investigate the design of KL-regularized policy gradient algorithms specifically for reasoning, Wu et al. (2024) revisit KL in the context of knowledge distillation for LLMs, and Vassoyan et al. (2025) argue that ignoring KL penalties on critical tokens can boost exploration in RL fine-tuning. Tang & Munos (2025) further analyze pitfalls in gradient estimation. All these studies suggest that while KL constraints are effective safeguards, their implications for reasoning insufficiently understood, motivating our investigation.

B Experimental setup - Further details

For the RL finetuning, we set the learning rate to 10^{-6}, number of rollouts per prompt $K = 5$, maximum response length to 1024, $temperature = 1.0$. We RL finetune the models on 2 GPU nodes consisting of 4 A100s (80GB) each using `verl` (Sheng et al., 2025). 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 §4.1 can be written as

$$D_{\mathrm{KL}}(A\|B) = \mathbb{E}_{Y\sim A}\Big[\log A_\theta(Y) - \log B_\phi(Y)\Big] \tag{19}$$

$$\frac{\partial}{\partial a} D_{\mathrm{KL}}(A\|B) = \mathbb{E}_{Y\sim A}\Big[\sum_{t=1}^{T}(y_t - p_t^A)\ (\log A_\theta(Y) - \log B_\phi(Y))\Big], \tag{20}$$

$$\frac{\partial}{\partial b} D_{\mathrm{KL}}(A\|B) = \mathbb{E}_{Y\sim A}\Big[\sum_{t=1}^{T}(y_t - p_t^A)\, c_{t-1}\ (\log A_\theta(Y) - \log B_\phi(Y))\Big]. \tag{21}$$

D Further empirical analysis

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

D.1 Examples where use of KL becomes necessary

As stated in §4.2, $\beta = 0$ may not always lead to the best performance, even within the domain of RLVR. While RL fine-tuning Qwen2.5-7B-Instruct on MATH train set, we observe that while the performance on MATH test set (nearly in-distribution to the training data) improves (albeit marginally) as compared to the base model, when training with $\beta = 0$, it drops, significantly in some cases on the out-of-distribution tasks. However, a KL penalty with $\beta = 0.05$ alleviates this performance degradation significantly shown in Fig. 7

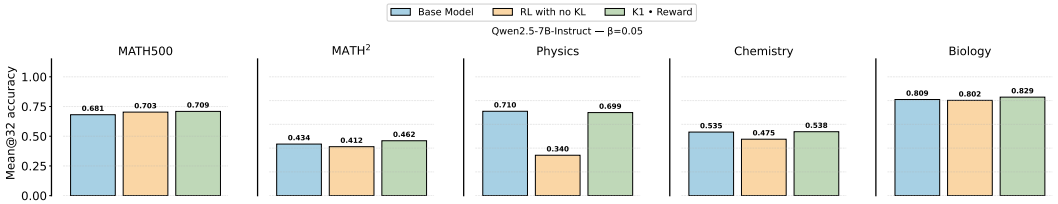

Figure 7: Including a KL penalty prevents during RL fine-tuning of Qwen2.5-7B-Instruct prevents performance degradation on OOD.

D.2 Evaluation results for $\beta = 0.3$ and 1

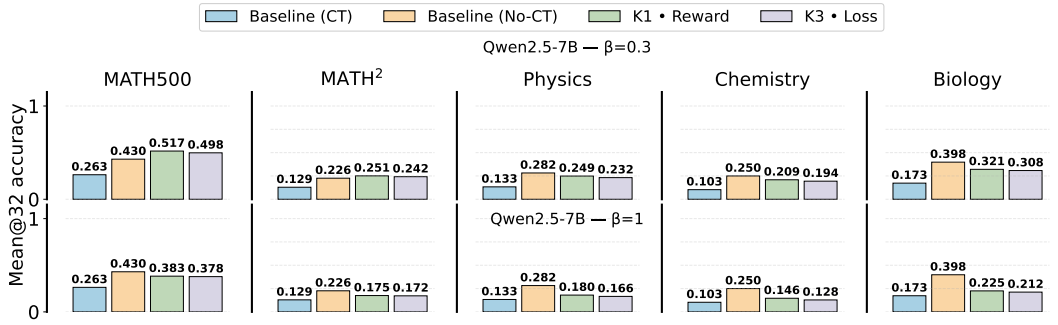

Figure 8: **Comparison of Qwen2.5-7B trained with two stable estimator configurations - K1 in reward and K3 in loss.** Baseline (CT) refers to the performance of base Qwen2.5-7B when prompted with a chat template. Baseline (No-CT) represents the performance when it is prompted with a chat template. K1 in Loss (unbiased gradient performs the beats on both in-domain and out-of-domain tasks. Increasing β consistently deteriorates performance.

E 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 2: Notation table.

E.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:

$$\nabla_\theta \mathbb{KL}(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\text{ref}}(\cdot \mid x)) = \nabla_\theta \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\log \frac{\pi_\theta(y_{1:T} \mid x)}{\pi_{\text{ref}}(y_{1:T} \mid x)} \right] \tag{22}$$

$$= \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\log \frac{\pi_\theta(y_{1:T} \mid x)}{\pi_{\text{ref}}(y_{1:T} \mid x)} \nabla_\theta \log \pi_\theta(y_{1:T} \mid x) \right]. \tag{23}$$

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

E.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.

$$\nabla_\theta \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\widehat{\text{KL}} \right] \tag{24}$$

$$= \nabla_\theta \sum_{y_{1:T}} \widehat{\text{KL}} \cdot \pi_\theta(y_{1:T} \mid x) \tag{25}$$

$$= \sum_{y_{1:T}} \left(\nabla_\theta \widehat{\text{KL}} \right) \cdot \pi_\theta(y_{1:T} \mid x) + \sum_{y_{1:T}} \widehat{\text{KL}} \cdot (\nabla_\theta \pi_\theta(y_{1:T} \mid x)) \tag{26}$$

$$= \underbrace{\mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \nabla_\theta \left[\widehat{\text{KL}} \right]}_{\text{path-wise derivative}} + \underbrace{\mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\widehat{\text{KL}} \cdot \nabla_\theta \log \pi_\theta(y_{1:T} \mid x) \right]}_{\text{score function derivative}} \tag{27}$$

$$= \underbrace{\mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\sum_t \nabla_\theta \widehat{\text{KL}}_t \right]}_{\text{Pathwise}} + \underbrace{\mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)} \left[\left(\sum_t \widehat{\text{KL}}_t \right) \cdot \nabla_\theta \log \pi_\theta(y_{1:T} \mid x) \right]}_{\text{Score function}}, \tag{28}$$

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.

E.3 K1 estimator

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

$$\text{K1} = \sum_{t=1}^{T} \text{K1}_t = \sum_{t=1}^{T} \log \frac{\pi_\theta(y_t \mid x, y_{<t})}{\pi_{\text{ref}}(y_t \mid x, y_{<t})}. \tag{29}$$

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

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.

$$\mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\nabla_\theta \sum_t \mathsf{K1}_t\right] = 0. \tag{30}$$

Score function derivative. The score function derivative of K1 is an unbiased estimate of the true gradient in Equation (23).

$$\begin{aligned}&\mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\left(\sum_t \mathsf{K1}_t\right)\cdot\nabla_\theta \log \pi_\theta(y_{1:T}\mid x)\right]\\ &\quad= \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\log\frac{\pi_\theta(y_{1:T}\mid x)}{\pi_{\mathrm{ref}}(y_{1:T}\mid x)}\nabla_\theta \log \pi_\theta(y_{1:T}\mid x)\right].\end{aligned} \tag{31}$$

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.

E.4 K3 ESTIMATOR

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

$$\mathsf{K3} = \sum_{t=1}^{T}\mathsf{K3}_t = \sum_{t=1}^{T}\left(\frac{\pi_{\mathrm{ref}}(y_t\mid y_{<t},x)}{\pi_\theta(y_t\mid y_{<t},x)} - 1 - \log\frac{\pi_{\mathrm{ref}}(y_t\mid y_{<t},x)}{\pi_\theta(y_t\mid y_{<t},x)}\right). \tag{32}$$

We first show that K3 is an unbiased estimator of $D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}})$:

$$\mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\mathsf{K3}\right] \tag{33}$$

$$= \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\sum_t\left(\frac{\pi_{\mathrm{ref}}(y_t\mid x,y_{<t})}{\pi_\theta(y_t\mid x,y_{<t})} - 1 - \log\frac{\pi_{\mathrm{ref}}(y_t\mid x,y_{<t})}{\pi_\theta(y_t\mid x,y_{<t})}\right)\right] \tag{34}$$

$$= \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\sum_t\frac{\pi_{\mathrm{ref}}(y_t\mid x,y_{<t})}{\pi_\theta(y_t\mid x,y_{<t})}\right] - T + \mathbb{E}_{y_{1:T}\sim\pi_\theta}\left[\log\frac{\pi_\theta(y_{1:T}\mid x)}{\pi_{\mathrm{ref}}(y_{1:T}\mid x)}\right] \tag{35}$$

$$= D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}}) \tag{36}$$

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 Equation (23).

$$\mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\nabla_\theta \mathsf{K3}_t\right] \tag{37}$$

$$= \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\nabla_\theta \sum_t \left(\frac{\pi_{\text{ref}}(y_t \mid x, y_{<t})}{\pi_\theta(y_t \mid x, y_{<t})} - 1 - \log \frac{\pi_{\text{ref}}(y_t \mid x, y_{<t})}{\pi_\theta(y_t \mid x, y_{<t})}\right)\right] \tag{38}$$

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

$$= -\mathbb{E}_{y_{<t}\sim\pi_\theta(\cdot|x), y_t\sim\pi_{\text{ref}}(\cdot|x,y_{<t})}\left[\sum_t \nabla_\theta \log \pi_\theta(y_t \mid x, y_{<t})\right] \tag{39}$$

$$= \mathbb{E}_{y_{<t}\sim\pi_\theta(\cdot|x)}\left[\sum_t \nabla_\theta \mathrm{KL}(\pi_{\text{ref}}(\cdot \mid x, y_{<t}) \,\|\, \pi_\theta(\cdot \mid x, y_{<t}))\right]. \tag{40}$$

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

Score function derivative. The score function derivative of K3 also is a biased estimate of the true gradient in Equation (23).

$$\mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\left(\sum_{t=1}^{T} \mathsf{K3}_t\right) \cdot \nabla_\theta \log \pi_\theta(y_{1:T} \mid x)\right]$$

$$= \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\left(\sum_t \left(\frac{\pi_{\text{ref}}(y_t \mid x, y_{<t})}{\pi_\theta(y_t \mid x, y_{<t})} - 1 - \log \frac{\pi_{\text{ref}}(y_t \mid x, y_{<t})}{\pi_\theta(y_t \mid x, y_{<t})}\right)\right) \nabla_\theta \log \pi_\theta(y_{1:T} \mid x)\right] \tag{41}$$

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

$$+ \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\log \frac{\pi_\theta(y_{1:T} \mid x)}{\pi_{\text{ref}}(y_{1:T} \mid x)} \nabla_\theta \log \pi_\theta(y_{1:T} \mid x)\right] \tag{42}$$

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

$$+ \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\log \frac{\pi_\theta(y_{1:T} \mid x)}{\pi_{\text{ref}}(y_{1:T} \mid x)} \nabla_\theta \log \pi_\theta(y_{1:T} \mid x)\right] \tag{43}$$

$$= -\mathbb{E}_{y_{<t}\sim\pi_\theta(\cdot|x)}\left[\sum_t \nabla_\theta \mathrm{KL}(\pi_{\text{ref}}(\cdot \mid x, y_{<t}) \,\|\, \pi_\theta(\cdot \mid x, y_{<t}))\right] + \mathbb{E}_{y_{1:T}\sim\pi_\theta(\cdot|x)}\left[\mathsf{K1} \cdot \nabla_\theta \log \pi_\theta(y_{1:T} \mid x)\right]. \tag{44}$$

where in Equation 43 the terms corresponding to $s < t$ and $s > t$ reduce to 0. The first term in Equation (44) 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.