DEMYSTIFYING TRAINING-INFERENCE MISMATCH IN LLM REINFORCEMENT LEARNING VIA A ZERO-MISMATCH REFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern LLM RL systems separate rollout generation from policy optimization. These two stages are expected to produce token probabilities that match exactly. However, implementation differences can make them assign different values to the same sequence under the same model weights, inducing Training–Inference Mismatch (TIM). TIM is difficult to inspect because it is entangled with off-policy drift and common stabilization mechanisms. In this work, we isolate TIM in a zero-mismatch diagnostic setting (*ExactRL*), and show that small token-level numerical disagreements can independently cause training collapse. We further show that TIM changes the effective optimization problem, and identify a set of remedies that could mitigate TIM. Our results suggest that TIM is not benign numerical noise, but a systems-level perturbation that should be treated as a first-order factor in analyzing LLM RL stability.

1 INTRODUCTION

Large language model (LLM) reinforcement learning (RL) has become a central paradigm for post-training foundation models and a key driver of recent advances in complex reasoning capabilities (Schulman et al., 2017; Ouyang et al., 2022; Ziegler et al., 2019; Shao et al., 2024; Stiennon et al., 2020; Liu et al., 2025b). However, RL training remains difficult to stabilize in practice: policies may rapidly degrade, causing reward signals to drop over short training windows.

Understanding what causes these collapses is therefore essential for building reliable LLM RL training systems (Sheng et al., 2025b; Hu et al.; Fu et al., 2025b; RadixArk Team, 2025; Cao et al., 2025; Sheng et al., 2025a; MiniMax et al., 2025). However, diagnosing the root cause is difficult because many failure modes are deeply entangled. A collapse may be caused by many factors, such as poorly tuned hyperparameters, reward misspecification, and reward hacking (Fu et al., 2025a; Pan et al., 2024). Among these factors, Training-Inference Mismatch (TIM) is an infrastructure-level confounder: numerical differences between training and inference engines can cause divergent token probabilities, even for the same input and model weights.

In response to these stability challenges, the community has developed a range of training-level stabilization techniques, including importance sampling, rejection sampling, and other forms of conservative policy updates (Schulman et al., 2017; Yao et al., 2025; Li et al., 2026; Ling Team et al., 2025; Zheng et al., 2025a; Liu et al., 2025a). Although effective in some settings, their connection to specific failure mechanisms remains unclear: the same technique may correct PPO mini-step off-policy drift, suppress TIM-induced numerical outliers, or introduce additional optimization bias. Without a TIM-free diagnostic baseline to isolate these effects, practitioners must tune interventions and filtering thresholds by trial and error rather than by causal diagnosis.

In this paper, we aim to systematically understand the impact of TIM on LLM RL stability. Specifically, we aim to answer two key questions: First, does TIM contribute to RL training instability, and if so, to what extent? Second, how do common stabilization techniques interact with TIM, what aspects of the mismatch do they mitigate, and what optimization side effects do they introduce?

To answer these questions, we develop *ExactRL*, a lightweight rollout engine that achieves zero-mismatch with FSDP (Zhao et al., 2023; Rajbhandari et al., 2020) engine on top of VeRL (Sheng

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

et al., 2025b). *ExactRL* eliminates TIM by unifying kernel and model implementations with the FSDP training engine, and by employing batch-invariant kernels (He & Thinking Machines Lab, 2025) (§ 3.1). Using *ExactRL*, we conduct fine-grained diagnostic studies of LLM RL stability. Concretely, our contributions are:

Isolating TIM impact for LLM RL: Using our TIM-free baseline, we identify TIM alone as a significant factor in triggering RL training collapse (§ 3.2).

Analyzing failure modes of TIM-induced RL training collapse. We then conduct ablation studies on TIM's role in RL training collapse under general setups. Specifically, we analyze why RL training collapse in both trainer-side log-probabilities recomputation and rollout-side log-probabilities bypass. We find that TIM fundamentally changes the optimization objective, thereby inducing distinct failure (§ 4.1).

Ablating effectiveness of algorithmic TIM compensation. Furthermore, we evaluate whether common stabilization techniques can effectively mitigate TIM, including truncated importance sampling (TIS) (Yao et al., 2025), and rejection sampling (RS) (Li et al., 2026)(§ 4.2). Based on our ablation study, we identify an effective combination of existing algorithmic TIM compensations that can closely track our TIM-free baseline.

2 Training-Inference Mismatch in LLM RL

Due to the implementation differences between the training engines (FSDP (Zhao et al., 2023; Rajbhandari et al., 2020), Megatron (Shoeybi et al., 2020), etc) and the inference engines (vLLM (Kwon et al., 2023), SGLang (Zheng et al., 2024), etc), including divergent model/kernel implementations, the probability distribution on the vocabulary for the next token can be different even with the exact same model checkpoint and inputs. This introduces an unintended off-policy bias between the sampling and model update. Different from the off-policy bias introduced by PPO mini-steps (Schulman et al., 2017), TIM off-policy bias is an infrastructure-level noise, which cannot be addressed by naive PPO clipping methods (discussed in §4).

We formulate this issue in RL objectives as follows: Given a context x and a sampled response $y = (a_1, \ldots, a_T)$, let $s_t = (x, y_{<t})$. We distinguish three token-level distributions: $\pi_\theta(a_t|s_t)$ denotes the current policy being optimized; $\pi_{old}^{\text{rollout}}(a_t|s_t)$ denotes the behavioral distribution realized by the rollout engine when the token is sampled; and $\pi_{old}^{\text{train}}(a_t|s_t)$ denotes the trainer-side reference distribution used when an algorithm requires an old-policy probability.

In an exact on-policy implementation, the probability assigned to each sampled token should be consistent between rollout and training. TIM occurs when the rollout execution path and the trainer execution path assign different probabilities to the same token under the same model weights and sampled sequence. At the token level, this discrepancy can be written as

$$\delta_t = \log \pi_{old}^{\text{train}}(a_t|s_t) - \log \pi_{old}^{\text{rollout}}(a_t|s_t). \tag{1}$$

This definition is objective-agnostic: the mismatch exists before choosing whether the update is implemented with REINFORCE (Williams, 1992; Hu et al., 2025), PPO (Schulman et al., 2017), or GRPO (Shao et al., 2024; Yu et al., 2025).

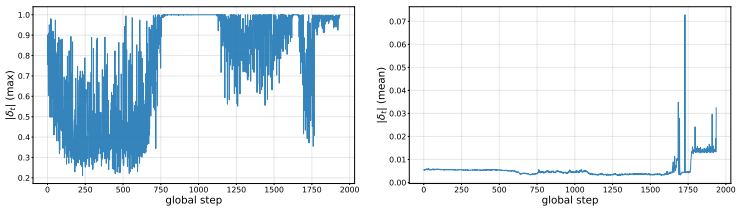
Table 1 illustrates an example of several sampled tokens from the same context and model checkpoint. For each token, we compare the log-probability produced by the rollout with that re-evaluated by the training engine. Ideally, these two values should be identical for an on-policy update. However, we can observe clear discrepancies in token log-probabilities between the two sides, including cases where the top-1 token choice at a given position is flipped. As Figure 1 shows, although the average difference in token-level probabilities is small per training batch, the maximum difference can even reach 1.0 for some extreme tokens, which is generally observable when TIM exists.

3 Isolating Training-Inference Mismatch with *ExactRL*

In this section, we isolate the impact of TIM on RL training stability. This requires two key ingredients: (1) First, we introduce a TIM-free rollout implementation, *ExactRL*, as a diagnostic baseline

Table 1: Token-level kernel-numerical drift between the rollout and training stacks on the same Qwen3-8B (bf16) weights, traced along one sentence from a greedy sampled response on an AIME-2024 problem. $\delta_t = \log \pi_{\text{train}} - \log \pi_{\text{rollout}}$. Most positions are bit-close, but † marks an argmax flip: at `that`, π_{train}'s top 1 token is actually the punctuation + newline string `":\n\n"` (log prob -0.577), so the training side would have ended the clause as "*The problem states:*\n\n".

	`The`	`problem`	`states`	`that`†	`there`	`exist`	`real`	`numbers`
$\log \pi_{\text{rollout}}$	-0.279	-0.063	-0.314	-0.694	-0.000	-0.030	-0.000	-0.000
$\log \pi_{\text{train}}$	-0.278	-0.063	-0.314	-0.827	-0.000	-0.038	-0.000	-0.000
δ_t	0.001	0.000	0.000	-0.133	0.000	-0.008	0.000	0.000



(a) $|\delta_t|$ (max) (b) $|\delta_t|$ (mean)

Figure 1: Statistical $|\delta_t|$ max and mean for every training batch in the Qwen3-1.7B GRPO experiment (detailed configuration in Appendix A.1). While the mean of $|\delta_t|$ is small, we can observe some extreme tokens with its $|\delta_t|$ near 1.0.

that removes infrastructure-induced mismatch from the RL loop. (2) Second, we evaluate this baseline under REINFORCE, which avoids PPO ratio clipping that may mask or distort TIM-induced changes in the loss and gradient signals.

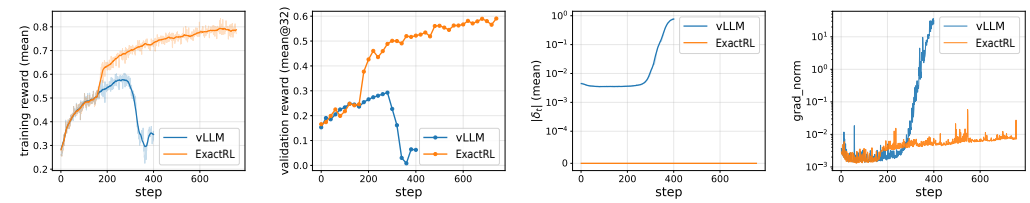
3.1 *ExactRL*: A Zero-mismatch Rollout Engine

For the TIM-free baseline, we introduce *ExactRL*, a lightweight rollout engine whose rollout token log-probabilities can achieve bit-wise alignment with the FSDP engine.

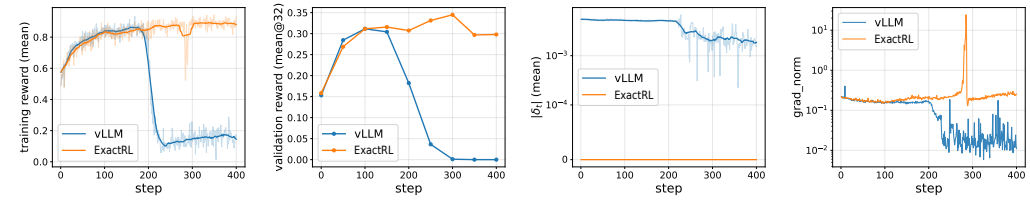
TIM comes from two sources: (1) The model and kernel implementation differences between the inference and training engines. Although semantically and mathematically the same, they often make different decisions regarding implementation details. For example, inference engines prefer inference-optimized kernel libraries like FlashInfer, which is not applicable in training engines. (2) Variations in kernel reduction order and tiling. Even when the same kernel implementation is used, performance-oriented optimizations such as atomic additions can introduce non-determinism, causing the kernel to produce different outputs for identical inputs. Moreover, even a *deterministic* kernel may exhibit batch-dependent numerical behavior: changes in batch size can trigger different launch-grid configurations through auto-tuning, thereby altering GPU tiling strategies and reduction orders. Since floating-point accumulation is non-associative under finite precision, these changes in execution order can ultimately lead to numerically different results.

Hence, *ExactRL* addresses these two sources of mismatch by (1) using the same HuggingFace-based model implementation and register *ExactRL* kernel implementation in the FSDP engine initialization and (2) employing deterministic and batch-invariant kernels, which fix the tiling and reduction order in the GPU kernel implementation. Following the original batch invariant kernel implementation (He & Thinking Machines Lab, 2025), *ExactRL* additionally implements RMSNorm (Zhang & Sennrich, 2019), batched matrix multiplication, and batch invariant Fused MoE kernels for efficient MoE training/inference. For attention implementation (Dao et al., 2022) specifically, we disable KV splitting (Dao et al., 2023) to ensure determinism as well.

Meanwhile, since fixed tiling in batch-invariant kernels trades performance for numerical stability, *ExactRL* retains reasonable throughput for practical RL training by integrating chunked prefill (Agrawal et al., 2023), CUDAGraph (NVIDIA Corporation, 2025), pipeline parallelism (Huang



(a) MoE training reward (b) MoE val-reward (c) MoE δ_t (mean) (d) MoE gradient norm



(e) Dense training reward (f) Dense val-reward (g) Dense δ_t (mean) (h) Dense gradient norm

Figure 2: REINFORCE experiments comparing vLLM non-exact rollout with *ExactRL*. Top row: Qwen3-30B-A3B MoE. Bottom row: Qwen3-1.7B dense. Each row reports training reward, AIME 2024 validation reward, δ_t (mean), and log-scale gradient norm. More results in Appendix A.2.

et al., 2019), and optimistic KV allocation with preemption fallback. *ExactRL* maintains hackable and very lightweight and its LOC is similar to nano-vLLM (GeeeekExplorer, 2025) .

3.2 Exposing the Impact of Training-Inference Mismatch

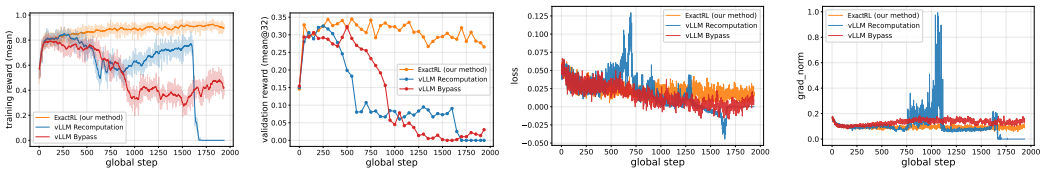
To isolate how TIM impacts RL training stability from low-level numerical disagreement, we first study REINFORCE (Williams, 1992) on-policy updates. Unlike PPO-style objectives, REINFORCE consumes each rollout batch in a single policy-gradient update. This makes it a cleaner diagnostic objective for attributing changes in loss, gradient, and reward to rollout-training probability disagreement.

We conduct REINFORCE experiments (with batch-whitened advantages) for both dense (Qwen3-1.7B) and MoE (Qwen3-30B-A3B) models (Yang et al., 2025). Both settings compare a standard non-exact rollout engine (vLLM) against *ExactRL*. The dense run is trained on Sanity-Test-R1D-1.5B (Qi et al., 2025) and evaluated on AIME 2024 (Mathematical Association of America, 2024) every 50 global steps. The MoE run is trained on DAPO (Yu et al., 2025) dataset and evaluated on AIME 2024 every 20 global steps. The detailed configuration is summarized in Appendix Table 2.

Figure 2 shows that the non-exact rollout run exhibits instability jointly in reward and gradient signals, while *ExactRL* is significantly more stable. For example, in the MoE setting of Figure 2, the vLLM run initially improves but starts to degrade after step 280, with training and validation rewards decreasing from 0.574 and 0.293 to 0.255 and 0.067, respectively. By contrast, the *ExactRL* reference remains stable and continues to improve, reaching 0.753 training reward and 0.534 validation reward. Since TIM is the only difference between *ExactRL* and vLLM baseline, *these experiments confirm that TIM is itself a critical destabilizing factor, not merely a secondary artifact compounded with other training effects*.

4 Ablating TIM Mitigations under the *ExactRL* Baseline

§ 3 shows that TIM alone can destabilize RL training with the REINFORCE setup. However, algorithms like PPO and GRPO have more complicated setups with PPO mini-steps. Different from fully on-policy algorithms like REINFORCE, algorithms like PPO and GRPO fix a snapshot of the policy (θ_{old}) to collect a batch of samples and perform multiple gradient steps on this batch. To optimize the current policy θ against the data sampled from a different distribution θ_{old}, they apply *importance sampling*, introducing the probability ratio π_θ/π_{old} into the objective.



(a) Training reward. (b) AIME24 val-reward. (c) Loss. (d) Gradient norm.

Figure 3: Qwen3-1.7B GRPO experiments with *ExactRL* and vLLM recomputation and bypass, where only *ExactRL* can maintain the training stability. Additional experimental results using the DAPO dataset, Llama models, and different random seeds are provided in Appendix A.3.

Under such setups, TIM is intertwined with stabilization techniques for them like PPO clipping. We now turn to the second research question: when practitioners apply common stabilization techniques, among which some are designed for general PPO training stability and some are TIM-aware, what do these mechanisms actually fix, and what optimization side effects do they introduce?

Experiment setup. We study this question in a practical GRPO training setting. Unless otherwise stated, the experiments in this section use Qwen3-1.7B with FSDP on mathematical reasoning workloads. The model is trained on Sanity-Test-R1D-1.5B and evaluated on AIME 2024 every 50 global steps. We compare *ExactRL* with vLLM non-exact rollout implementations using recomputation, bypass, rollout correction, and rejection sampling variants.

4.1 The Failure Modes of Recomputation and Bypass

There are two implementations for acquiring π_{old}: recomputation (where the trainer re-evaluates the sampled tokens) and bypass (where the rollout engine directly transmits its log-probabilities). Both strategies use the same clipped PPO/GRPO surrogate, but differ in how the denominator of the policy ratio is obtained. For a sampled token a_t under state s_t with advantage A_t, the token-level clipped surrogate is

$$\mathcal{L}_{\text{ppo}}(r_{ppo}, A) = -\min\left(r_{ppo}A, \text{clip}(r_{ppo}, 1-\epsilon, 1+\epsilon)A\right). \tag{2}$$

Under recomputation and bypass, the PPO ratios are respectively defined as

$$r_{ppo}^{\text{train}} = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{old}^{\text{train}}(a_t \mid s_t)}, \qquad r_{ppo}^{\text{rollout}} = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{old}^{\text{rollout}}(a_t \mid s_t)}. \tag{3}$$

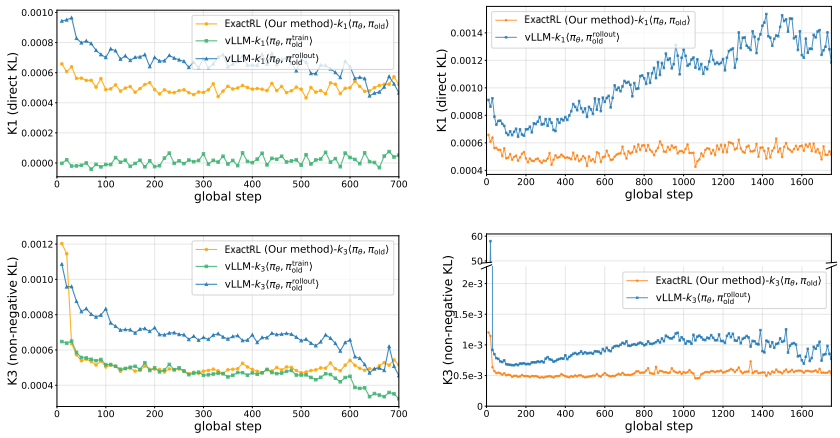
Thus, recomputation and bypass instantiate the same clipped surrogate with different denominators:

$$\mathcal{L}_{\text{recomp}} = \mathcal{L}_{\text{ppo}}(r_{ppo}^{\text{train}}, A), \qquad \mathcal{L}_{\text{bypass}} = \mathcal{L}_{\text{ppo}}(r_{ppo}^{\text{rollout}}, A). \tag{4}$$

The sequence-level training loss is obtained by summing these token-level terms over the response and averaging over the sampled batch.

Figure 3 shows the main phenomenon: while *ExactRL* keeps the training reward around 0.93, vLLM recomputation first degrades from about 0.87 to roughly 0.40 during the first 650 steps, partially recovers afterward, and then drops rapidly again after about step 1610 before collapsing to near-zero reward after about step 1665. vLLM bypass shows a single-stage degradation, with training reward dropping to roughly 0.4 but not collapsing to zero, and this degradation is not accompanied by comparably large loss spikes. Unlike the REINFORCE setting in § 3.2, where reward collapse is accompanied by clear loss and gradient-norm anomalies, the GRPO runs in Figure 3 show less synchronized behavior across these signals: bypass exhibits reward degradation without a comparably synchronized loss anomaly, while recomputation enters an early reward-degradation phase before the later gradient-norm spike becomes visible. This motivates a finer-grained analysis.

KL estimators are not sufficient indicators. Since TIM directly perturbs the effective distance between the updated policy π_θ and the old policy π_{old}, we first inspect KL estimators computed on the PPO ratio (Schulman, 2020), such as $K_1(r_{\text{ppo}}) = -\log r_{\text{ppo}}$ and $K_3(r_{\text{ppo}}) = (r_{\text{ppo}} - 1) - \log r_{\text{ppo}}$, where r_{ppo} can be instantiated as either $r_{\text{ppo}}^{\text{train}}$ or $r_{\text{ppo}}^{\text{rollout}}$ as defined in Eq. 3. In bypass mode, where the rollout and trainer probabilities become visibly inconsistent, these aggregate KL



(a) Recomputation: K_1/K_3 metrics (b) Bypass: K_1/K_3 metrics

Figure 4: KL estimators under recomputation and bypass. In recomputation mode, both K_1 and K_3 remain nearly flat during the first 700 steps, even though the reward is already entering the degradation phase. In bypass mode, both K_1 and K_3 increase noticeably. Under *ExactRL*, $\pi_{\text{old}}^{\text{train}} = \pi_{\text{old}}^{\text{rollout}}$, so the two corresponding estimators coincide. With recomputation, $K_1\langle\pi_\theta, \pi_{\text{old}}^{\text{train}}\rangle$ is lower than *ExactRL* because K_1 estimator is signed and each token's K_1 can cancel out when accumulating at the batch level.

probes increase noticeably in both K_1 and K_3. However, under recomputation mode, even over the first 700 training steps in which the run already exhibits the onset of failure, the KL estimators remain close to the *ExactRL* baseline and fail to expose the emerging instability. This indicates that aggregate probability-space divergence is not sufficient to characterize the earliest stage of TIM-induced training failure.

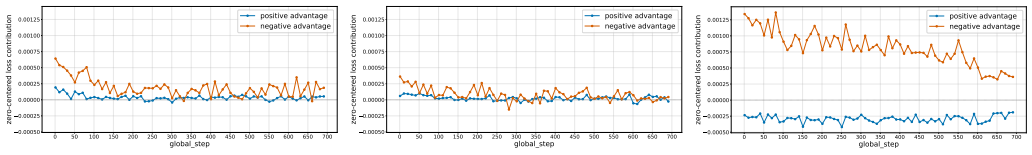
Zero-centered loss contribution. The preceding analysis characterizes TIM in probability space. We further study how such probability space change impacts policy update. First, considering that PPO loss applies different clipping boundaries to positive- and negative-advantage samples, we therefore analyze the two groups separately. Second, we subtract 1 from r_{ppo} to measure the loss change relative to $r_{\text{ppo}} = 1$, removing the advantage-dependent offset $-A_t$. We define the zero-centered loss contribution:

$$C(r_{ppo}) = -(r_{ppo} - 1)A_t, \qquad r_{ppo} \in \{r_{ppo}^{\text{train}}, r_{ppo}^{\text{rollout}}\}, \tag{5}$$

which has the same gradient as the standard $-r_{ppo}A_t$ objective but is zero when $r_{ppo} = 1$. We evaluate this quantity using two denominators for the same sampled token. $r_{\text{ppo}}^{\text{train}}$ uses the trainer-recomputed distribution π_{train} and $r_{\text{ppo}}^{\text{rollout}}$ uses the rollout distribution π_{rollout}. Comparing $C(r_{\text{ppo}}^{\text{train}})$ with $C(r_{\text{ppo}}^{\text{rollout}})$ therefore isolates how trainer-side recomputation alters the ratio-dependent surrogate contribution for the same sampled token. Under *ExactRL*, the two distributions are numerically aligned, yielding $r_{\text{ppo}}^{\text{train}} = r_{\text{ppo}}^{\text{rollout}}$ and hence identical contributions.

Figure 5 shows a clear sign-dependent distortion under TIM. With *ExactRL*, rollout- and trainer-referenced contributions remain aligned. With vLLM, the rollout-referenced contribution stays near zero, whereas the trainer-referenced contribution separates by advantage sign, with a larger positive shift for negative-advantage tokens. Thus, even when aggregate KL estimators remain close to the *ExactRL* baseline, recomputation can already introduce a systematic bias in the gradient-carrying update signal. Such biased updates can accumulate before the mismatch becomes visible as a clear KL spike.

Why bypass also fails. In bypass mode, the PPO ratio correctly uses the behavior distribution in the denominator ($r_{ppo}^{\text{rollout}} = \pi_\theta^{train}/\pi_{old}^{\text{rollout}}$). However, the numerator π_θ^{train} is evaluated using the trainer's numerical execution path. During the backward pass, the optimizer computes the score function gradient based on $\nabla_\theta \log \pi_\theta^{train}$. Because TIM creates a misaligned probability landscape between the trainer and the rollout engine, the optimizer exploits numerical artifacts in the trainer's



(a) *ExactRL* $C(r_{ppo}^{\text{rollout}})$/$C(r_{ppo}^{\text{train}})$ (b) vLLM $C(r_{ppo}^{\text{rollout}})$ (c) vLLM $C(r_{ppo}^{\text{train}})$

Figure 5: Zero-centered loss contribution $C(r) = -(r-1)A$, which isolates the ratio-dependent component of the surrogate objective. This shows that recomputation induces a sign-dependent skew in the advantage-weighted update signal, rather than merely adding uniform noise.

forward pass. These weight updates fail to translate into actual behavioral improvements when θ is deployed back to the rollout engine, leading to silent policy degradation. To validate this hypothesis, we conduct a minimal *model update direction validation* experiment, which computes logprobs of past trajectories after the model update. For the tokens in the first 10 steps of training Qwen3-1.7B on the GSM-8K dataset, 17.3% of token logprob change directions are misaligned with their advantage sign, while *ExactRL* can ensure all the token logprob update directions are aligned.

Takeaway. (1) In recomputation mode, the sampling log-probabilities used in loss computation are not from the actual samplers (rollout), resulting in a skew in the advantage-weighted loss contributions seen by the optimizer. (2) Even with bypass mode, the actual optimization target still exists in a different sampling space from the rollout, making policy update ineffective.

4.2 Ablating Algorithmic TIM Compensation

§ 4.1 shows that recomputation and bypass instantiate different ways of accounting for the old-policy probability, but neither removes TIM at the source. This motivates a natural question: can post-hoc algorithmic compensation recover the behavior of zero-mismatch rollout, or does it only suppress some observable symptoms? We answer this question using *ExactRL* as a TIM-free reference. Our goal is to diagnose what existing TIM-aware correction scaffolds can approximate, which design choices matter, and where their limits remain.

Existing algorithmic corrections. Existing rollout-correction methods commonly suppress unreliable samples through token-level truncation or sequence-level rejection. Let $r_{\text{corr}} = \frac{\pi_{\text{old}}^{\text{train}}}{\pi_{\text{old}}^{\text{rollout}}}$ denote the correction ratio. They typically act either at the token level truncated importance sampling,

$$\mathcal{L}_{\text{TIS}} = \sum_{t=1}^{T} \min(r_{\text{corr},t}, \tau_{\text{tok}}) \mathcal{L}_{\text{PPO}} \left(r_{\text{ppo},t}^{\text{train}}, A_t\right), \tag{6}$$

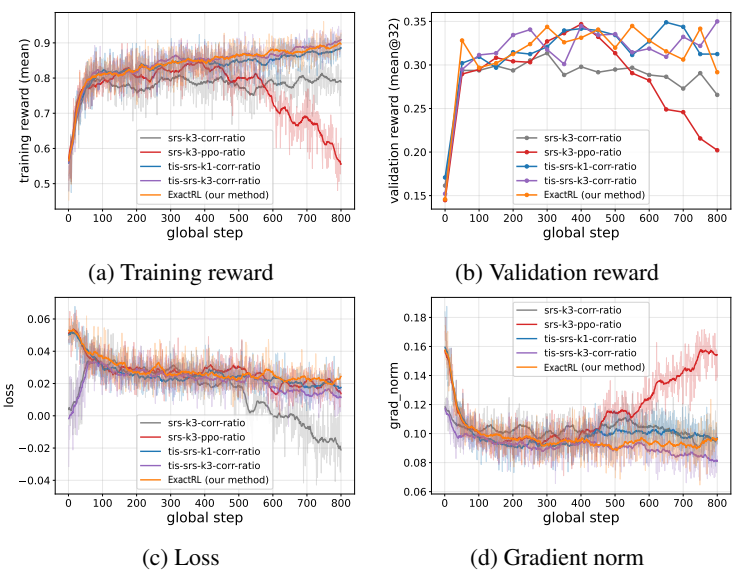
or at the sequence level through rejection sampling,

$$\mathcal{L}_{\text{RS}} = \mathbf{1}\left[S_{\text{seq}}(q_{1:T}) \le \tau_{\text{seq}}\right] \sum_{t=1}^{T} \mathcal{L}_{\text{PPO}} \left(r_{\text{ppo},t}^{\text{rollout}}, A_t\right). \tag{7}$$

Here, τ_{tok} is a token-level truncation threshold that limits large correction weights, and τ_{seq} is the sequence-level rejection threshold. For the specific threshold values, we follow prior work (Qiu et al., 2026; Li et al., 2026)(more details in Appendix A.4). $q_{1:T}$ denotes the sequence of token-level diagnostic signals used for rejection, $S_{\text{seq}}(\cdot)$ maps these signals to a scalar trajectory-level trust-region score.

Our diagnostic instantiation. We therefore use these post-hoc corrections as diagnostic probes rather than as new algorithmic proposals. § 4.1 shows that TIM can enter PPO/GRPO both through trainer–rollout old-policy displacement and through skewed token-level loss contributions. These observations motivate a controlled ablation along two axes: the ratio used to drive sequence-level rejection and the effect of adding token-level truncation before sequence-level rejection.

For the masking signal, we compare $q_t \in \{r_{\text{corr},t}, r_{\text{ppo},t}^{\text{rollout}}\}$. Here, $r_{\text{corr},t}$ measures system-induced mismatch between $\pi_{\text{old}}^{\text{train}}$ and $\pi_{\text{old}}^{\text{rollout}}$, whereas $r_{\text{ppo},t}^{\text{rollout}}$ measures policy movement from the rollout



(a) Training reward (b) Validation reward

(c) Loss (d) Gradient norm

Figure 6: Algorithmic-patch comparison. We evaluate four correction baselines: `srs-k3-corr-ratio` and `srs-k3-ppo-ratio` apply sequence-level rejection using r_{corr} and r_{ppo}, respectively, while `tis-srs-k1-corr-ratio` and `tis-srs-k3-corr-ratio` combine truncated importance sampling with r_{corr}-based sequence rejection instantiated with K1 and K3, respectively. We provide the expanded objectives in Appendix A.4. For sequence rejection, r_{corr} is more effective than r_{ppo}. Additionally with TIS, r_{corr}-based sequence rejection can track *ExactRL* closely. For better clarity, curves are smoothed by center mean with window size of 25.

behavior policy to the current policy. For both choices, q_t is used only as the masking signal, rather than necessarily as the PPO ratio inside the surrogate objective. At the granularity level, we apply sequence-level rejection to both signals by aggregating token-wise mismatch into $S_{\text{seq}}(q_{1:T}) = \sum_{t=1}^{T} K(q_t)$, with $K(\cdot)$ instantiated as K_1 or K_3, and rejecting trajectories whose accumulated mismatch exceeds a threshold. We evaluate a joint token- and sequence-level variant, which first filters localized token-level outliers and then applies the sequence-level rejection criterion.

Together, these choices yield three diagnostic configurations: sequence-level rejection based on either r_{corr} or $r_{\text{ppo}}^{\text{rollout}}$, and joint token-level clipping plus sequence-level rejection based on r_{corr}, with the sequence-level score instantiated by either $K1$ or $K3$. Objectives are detailed in Appendix A.4.

Experiment results. As Figure 6 shows, comparing the sequence-level variants, using r_t^{corr} as the filtering signal yields higher reward than using $r_{\text{ppo},t}^{\text{rollout}}$. We attribute this difference to the distinct semantics of the two ratios: The correction ratio clipping guarantees that the *base distribution (π_{old}^{train}) to update is not too far from the sampling space.* In contrast, the rollout ratio overlaps with PPO's policy-ratio mechanism, whose purpose is to control the update magnitude of how far the current policy moves away from the behavior policy, but not the starting distribution location.

The same ablation also shows that the configuration *combining token-level filtering with sequence-level rejection* most closely tracks the *ExactRL* reference, indicating that TIM manifests at multiple granularities. Localized token-level mismatch outliers can distort individual PPO contributions even when the aggregate sequence-level score remains moderate, while some trajectories exhibit large accumulated mismatch and should be rejected at the sequence level. The choice between $K1$ and $K3$ for the sequence rejection has a comparatively minor effect in our experiments.

These results suggest that algorithmic correction can closely approach the zero-mismatch reference in the evaluated setting. However, these forms of algorithmic TIM compensation remains post-hoc: unlike *ExactRL*, it can only suppress already-generated samples and may also discard useful learning signals. More importantly, without *ExactRL* as a ground-truth reference, patch configuration (e.g., τ_{seq}, τ_{tok}) remains largely ad hoc. *ExactRL* therefore complements algorithmic patches by enabling their principled calibration.

Takeaway. (1) r_{corr}-based sequence-level rejection sampling is more effective than r_{ppo}-based. The threshold metric choice of K_1 or K_3 does not affect much. (2) TIS is effective as it fixes the PPO ratio used in loss function from r_{ppo}^{train} to $r_{ppo}^{rollout}$ by multiplying r_{ppo}^{train} with r_{corr}, consistent with our conclusion from § 4.1. (3) Combining (1) and (2), we find that carefully designed algorithmic configurations can closely track our zero-mismatch reference *ExactRL*.

5 Related Works

RL stabilization techniques. Existing stabilization techniques for RL training fall into three categories. *Fine-grained clipping regions* such as DAPO (Yu et al., 2025) apply a non-symmetric clipping region for tokens with positive and negative advantages. GSPO (Zheng et al., 2025b) moves to sequence-level importance ratios to reduce variance accumulation over long responses; *TIM-aware corrections* explicitly target the training–inference numerical mismatch: Truncated IS (TIS) (Yao et al., 2025) truncates extreme token-level IS weights, and Masked IS (Zheng et al., 2025a; Li et al., 2026) suppresses highly divergent samples by zeroing gradient contributions at either the token or sequence level. Score-centering (Marek & Ryabinin, 2026) decomposes policy update gradients into a drift term caused by training-inference mismatch and a reward-covariance signal term, and removes the drift by subtracting the sampler's expected score, stabilizing off-policy RL without importance ratios. Additionally, (Qi et al., 2025) and (Dirhoussi et al., 2026) find that using FP16 instead of BF16 is helpful for RL training stability. *MoE-specific corrections* address the additional instability from dynamic expert routing. For example, R3 (Ma et al., 2025) directly replays inference-time routing decisions during training mini-steps to eliminate divergent expert selections. Unlike these TIM compensation mechanisms, *ExactRL* eliminates the root cause of the numerical mismatch.

Efficient batch-invariant kernels. To address the performance penalty of fixing tiling and reduction order in batch invariant kernels, DeepSeek-V4 (DeepSeek-AI, 2026) introduces dual-kernel strategies for the attention kernel and a set of optimizations for GEMM kernels. Tree-based invariant kernels (TBIK) (Zhang et al., 2025) resolve the accumulation order mismatch across tensor-parallel orders by using a tree-based reduction order, enabling zero mismatch rollout under TP inference. They prove that batch-invariant kernels can be highly performant and scalable for large-scale RL. Our work focuses more on the analysis of TIM's role in RL training collapse with *ExactRL* baseline.

6 Discussion and Limitations

Relationship with asynchronous LLM RL. Large-scale agentic RL setups are typically asynchronous to speed up training, which introduces policy staleness as another off-policy source. Concurrent work (Marek & Ryabinin, 2026) finds that corrections effective against policy staleness (e.g., PPO clipping) fail under numerical mismatch, indicating that the two are distinct sources of off-policy behavior that co-occur in asynchronous RL pipelines. *ExactRL* removes the numerical mismatch, allowing TIM to be isolated from staleness. Moreover, *ExactRL* allows practitioners to systematically tune algorithmic setups and sensitive filtering thresholds to address the off-policy issues introduced by staleness in a noise-free environment.

Limitations. For the convenience of comparison with algorithmic compensation methods, our experiments primarily focus on the math datasets, as these works also mainly report results on math datasets. We will evaluate broader tasks like coding in the future work.

7 Conclusion

We investigate Training-Inference Mismatch (TIM) as a systems-level confounder in LLM RL stability. With our zero-mismatch baseline *ExactRL*, we demonstrate that TIM alone can destabilize RL training under a wide range of setups. We further analyze how TIM alters the effective optimization problem, explaining why common implementation choices such as trainer-side recomputation and rollout-side bypass fail to fully eliminate its impact. Using *ExactRL* as a reference, we show that carefully configured combinations of existing algorithmic corrections can closely match its performance. Overall, our findings call for a joint system-algorithm perspective on RL stability and highlight the need for zero-mismatch RL execution.

AI USE STATEMENT

We used generative AI tools for manuscript organization, editing the paper to improve readability and language, and retrieving and identifying relevant literature. We did not use generative AI tools to develop theoretical models or conceptual frameworks, propose or refine hypotheses, design or provide feedback on research methodology. Generating synthetic datasets, translation, and qualitative or thematic data analysis are not applicable to this work. We reviewed all AI-assisted work: organizational and editorial suggestions were checked against the authors' intended meaning. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We are committed to ensuring the reproducibility of our results. We will publicly release the source code and scripts needed to reproduce the experiments in this work. The appendix provides the experimental configurations and hyperparameters used for the reported results. Together, the released code and the settings documented in the appendix will enable independent reproduction of our experiments and main findings.

REFERENCES

Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, and Ramachandran Ramjee. Sarathi: Efficient llm inference by piggybacking decodes with chunked prefills, 2023. URL https://arxiv.org/abs/2308.16369.

Shiyi Cao, Dacheng Li, Fangzhou Zhao, Shuo Yuan, Sumanth R. Hegde, Connor Chen, Charlie Ruan, Tyler Griggs, Shu Liu, Eric Tang, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Skyrl-agent: Efficient rl training for multi-turn llm agent, 2025. URL https://arxiv.org/abs/2511.16108.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: fast and memory-efficient exact attention with io-awareness. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Tri Dao, Daniel Haziza, Francisco Massa, and Grigory Sizov. Flash-decoding for long-context inference, 2023. URL https://crfm.stanford.edu/2023/10/12/flashdecoding.html. Blog post.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Amine Dirhoussi, Quentin Gallouédec, Edward Beeching, Lewis Tunstall, Kashif Rasul, and Leandro von Werra. Defeating the trainer-generator precision mismatch in trl, 2026.

Jiayi Fu, Xuandong Zhao, Chengyuan Yao, Heng Wang, Qi Han, and Yanghua Xiao. Reward shaping to mitigate reward hacking in RLHF. In *ICML 2025 Workshop on Reliable and Responsible Foundation Models*, 2025a. URL https://openreview.net/forum?id=62A4d5Mokc.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, et al. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *arXiv preprint arXiv:2505.24298*, 2025b.

GeeeekExplorer. nano-vllm: Nano vllm. https://github.com/GeeeekExplorer/nano-vllm, 2025. Accessed: 2026-04-09.

Horace He and Thinking Machines Lab. Defeating nondeterminism in llm inference. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250910. https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/.

Jian Hu, Xibin Wu, Wei Shen, Jason Klein Liu, Zilin Zhu, et al. Openrlhf: an easy-to-use, scalable and high-performance rlhf framework (2024). *arXiv preprint arXiv:2405.11143*.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. Reinforce++: Stabilizing critic-free policy optimization with global advantage normalization, 2025. URL https://arxiv.org/abs/2501.03262.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism, 2019. URL https://arxiv.org/abs/1811.06965.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Yingru Li, Jiacai Liu, Jiawei Xu, Yuxuan Tong, Ziniu Li, Qian Liu, and Baoxiang Wang. Trust region masking for long-horizon llm reinforcement learning, 2026. URL https://arxiv.org/abs/2512.23075.

Ling Team, Anqi Shen, Baihui Li, Bin Hu, Bin Jing, Cai Chen, Chao Huang, Chao Zhang, Chaokun Yang, Cheng Lin, Chengyao Wen, Congqi Li, Deng Zhao, Dingbo Yuan, Donghai You, Fagui Mao, Fanzhuang Meng, Feng Xu, Guojie Li, Guowei Wang, Hao Dai, Haonan Zheng, Hong Liu, Jia Guo, Jiaming Liu, Jian Liu, Jianhao Fu, Jiannan Shi, Jianwen Wang, Jianxin Lai, Jin Yang, Jun Mei, Jun Zhou, Junbo Zhao, Junping Zhao, Kuan Xu, Le Su, Lei Chen, Li Tang, Liang Jiang, Liangcheng Fu, Lianhao Xu, Linfeng Shi, Lisha Liao, Longfei Zheng, Meng Li, Mingchun Chen, Qi Zuo, Qiang Cheng, Qianggang Cao, Qitao Shi, Quanrui Guo, Senlin Zhu, Shaofei Wang, Shaomian Zheng, Shuaicheng Li, Shuwei Gu, Siba Chen, Tao Wu, Tao Zhang, Tianyu Zhang, Tianyu Zhou, Tiwei Bie, Tongkai Yang, Wang Hong, Wang Ren, Weihua Chen, Wenbo Yu, Wengang Zheng, Xiangchun Wang, Xiaodong Yan, Xiaopei Wan, Xin Zhao, Xinyu Kong, Xinyu Tang, Xudong Han, Xudong Wang, Xuemin Yang, Xueyu Hu, Yalin Zhang, Yan Sun, Yicheng Shan, Yilong Wang, Yingying Xu, Yongkang Liu, Yongzhen Guo, Yuanyuan Wang, Yuchen Yan, Yuefan Wang, Yuhong Guo, Zehuan Li, Zhankai Xu, Zhe Li, Zhenduo Zhang, Zhengke Gui, Zhenxuan Pan, Zhenyu Huang, Zhenzhong Lan, Zhiqiang Ding, Zhiqiang Zhang, Zhixun Li, Zhizhen Liu, Zihao Wang, and Zujie Wen. Every step evolves: Scaling reinforcement learning for trillion-scale thinking model, 2025. URL https://arxiv.org/abs/2510.18855.

Jiacai Liu, Yingru Li, Yuqian Fu, Jiawei Wang, Qian Liu, and Yu Shen. When speed kills stability: Demystifying RL collapse from the training-inference mismatch, sep 2025a. URL https://richardli.xyz/rl-collapse.

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

Wenhan Ma, Hailin Zhang, Liang Zhao, Yifan Song, Yudong Wang, Zhifang Sui, and Fuli Luo. Stabilizing moe reinforcement learning by aligning training and inference routers. *arXiv preprint arXiv:2510.11370*, 2025.

Martin Marek and Max Ryabinin. Score centering stabilizes off-policy reinforcement learning, 2026. URL https://arxiv.org/abs/2609.20807.

Mathematical Association of America. AIME 2024: American Invitational Mathematics Examination, 2024. URL https://artofproblemsolving.com/wiki/index.php/2024_AIME_I.

MiniMax, Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.

NVIDIA Corporation. *CUDA Programming Guide: CUDA Graphs*, 2025. Accessed: 2026-04-09.

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. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Alexander Pan, Erik Jones, Meena Jagadeesan, and Jacob Steinhardt. Feedback loops with language models drive in-context reward hacking. *arXiv preprint arXiv:2402.06627*, 2024.

Penghui Qi, Zichen Liu, Xiangxin Zhou, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Defeating the training-inference mismatch via FP16. *arXiv preprint arXiv:2510.26788*, 2025. URL https://arxiv.org/abs/2510.26788.

Zhaopeng Qiu, Shuang Yu, Jingqi Zhang, Shuai Zhang, Xue Huang, Jingyi Yang, and Junjie Lai. FP8-RL: A practical and stable low-precision stack for LLM reinforcement learning. *arXiv preprint arXiv:2601.18150*, 2026.

RadixArk Team. Introducing miles: Rl framework to fire up large-scale moe training. https://lmsys.org/blog/2025-11-19-miles/, 2025.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models, 2020. URL https://arxiv.org/abs/1910.02054.

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

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

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

Guangming Sheng, Yuxuan Tong, Borui Wan, Wang Zhang, Chaobo Jia, Xibin Wu, Yuqi Wu, Xiang Li, Chi Zhang, Yanghua Peng, et al. Laminar: A scalable asynchronous rl post-training framework. *arXiv preprint arXiv:2510.12633*, 2025a.

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*, EuroSys '25, pp. 1279–1297. ACM, March 2025b. doi: 10.1145/3689031.3696075. URL http://dx.doi.org/10.1145/3689031.3696075.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020. URL https://arxiv.org/abs/1909.08053.

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. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Feng Yao, Liyuan Liu, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. On the rollout-training mismatch in modern RL systems. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025. URL https://openreview.net/forum?id=8MHqvb4lK9.

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. URL https://arxiv.org/abs/2503.14476.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, pp. 12360–12371, 2019. URL https://arxiv.org/abs/1910.07467.

Ziyang Zhang, Xinheng Ding, Jiayi Yuan, Rixin Liu, Huizi Mao, Jiarong Xing, and Zirui Liu. Deterministic inference across tensor parallel sizes that eliminates training-inference mismatch, 2025. URL https://arxiv.org/abs/2511.17826.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. Pytorch fsdp: Experiences on scaling fully sharded data parallel, 2023. URL https://arxiv.org/abs/2304.11277.

Chujie Zheng, Kai Dang, Bowen Yu, Mingze Li, Huiqiang Jiang, Junrong Lin, Yuqiong Liu, Hao Lin, Chencan Wu, Feng Hu, An Yang, Jingren Zhou, and Junyang Lin. Stabilizing reinforcement learning with llms: Formulation and practices, 2025a. URL https://arxiv.org/abs/2512.01374.

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

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs, 2024. URL https://arxiv.org/abs/2312.07104.

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. In *arXiv preprint arXiv:1909.08593*, 2019.

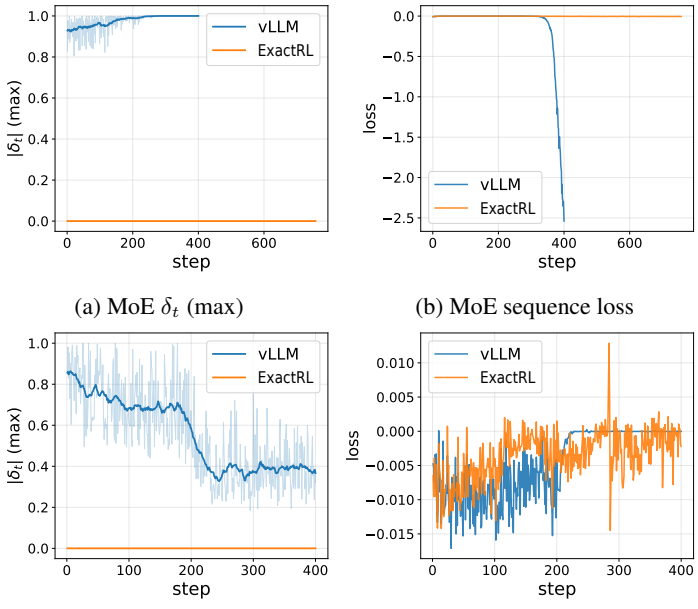
A Additional Details

A.1 Experimental Settings

Table 2: Experimental settings corresponding to the dense GRPO, dense REINFORCE, and MoE REINFORCE recipes.

Setting	**Dense GRPO**	**Dense REINFORCE**	**MoE REINFORCE**
Model	Qwen3-1.7B, Llama-3.2-1B	Qwen3-1.7B	Qwen3-30B-A3B
RL algorithm	GRPO	REINFORCE	REINFORCE
Training data	Sanity-Test-R1D-1.5B	Sanity-Test-R1D-1.5B	DAPO-Math-17k
Evaluation	AIME 2024 every 50 steps	AIME 2024 every 50 steps	AIME 2024 every 20 steps
Batching	Global batch 64, mini-batch 16, rollout group 8	Global batch 64	Global batch 512
Sequence length	Prompt 1024, response 8192	Prompt 1024, response 8192	Prompt 2048, response 20480
Hardware	1 node, 8 GPUs (H100)	2 nodes, 16 GPUs (H100)	8 nodes, 64 GPUs (H100)
Engine	FSDP2 + vLLM/*ExactRL*	FSDP2 + vLLM/*ExactRL*	FSDP2 + vLLM/*ExactRL*

A.2 Additional REINFORCE Results



(a) MoE δ_t (max)

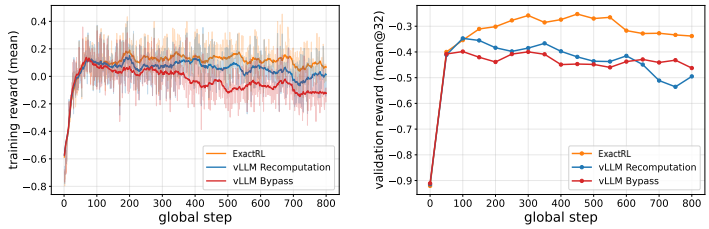
(b) MoE sequence loss

(c) Dense δ_t (max)

(d) Dense sequence loss

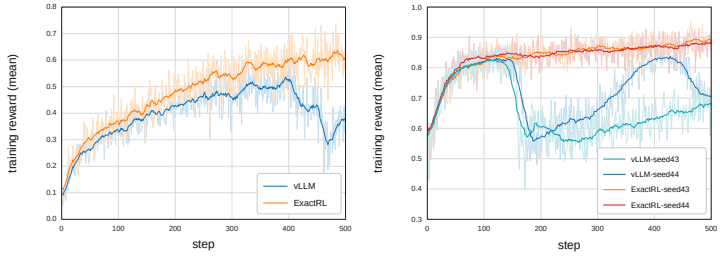
Figure 7: Additional metrics of REINFORCE experiments comparing vLLM non-exact rollout with *ExactRL*. Each row reports δ_t (max), and sequence loss. Top row: Qwen3-30B-A3B MoE. Bottom row: Qwen3-1.7B dense.

A.3 Additional Experiment Results



(a) Training reward. (b) Validation reward.

Figure 8: Qwen3-1.7B GRPO experiments on the DAPO dataset, with *ExactRL* and vLLM recomputation and bypass, where *ExactRL* can maintain the training stability better.



(a) Comparison on Llama 3.2 model. (b) Comparison on different seeds.

Figure 9: Additional GRPO experiments on Llama 3.2 and across different random seeds, comparing *ExactRL* with vLLM. All preceding experiments use random seed 42, we additionally report results with seeds 43 and 44 here. Across all seeds, the results support the same conclusion as in the paper: *ExactRL* consistently maintains stable training.

A.4 Expanded Objectives for Post-hoc Patch Variants in Section 4.2

Let $r_{\text{ppo},t}^{\text{rollout}}$ be the rollout-side PPO ratio defined in Eq. 3, and let $\mathcal{L}_{\text{PPO}}$ denote the token-level clipped surrogate.

srs-k3-corr-ratio. Sequence-level rejection using the correction ratio r_{corr} as the filtering signal:

$$\mathcal{L}_{\text{srs-k3-corr}} = \mathbf{1}\left[\sum_{t=1}^{T} K3(r_{\text{corr},1:T}) \le \tau_{\text{seq}}\right] \sum_{t=1}^{T} \mathcal{L}_{\text{PPO}}\left(r_{\text{ppo},t}^{\text{rollout}}, A_t\right). \tag{8}$$

srs-k3-ppo-ratio. Sequence-level rejection using the rollout-side PPO ratio $r_{\text{ppo}}^{\text{rollout}}$ as the filtering signal:

$$\mathcal{L}_{\text{srs-k3-ppo}} = \mathbf{1}\left[\sum_{t=1}^{T} K3(r_{\text{ppo},1:T}^{\text{rollout}}) \le \tau_{\text{seq}}\right] \sum_{t=1}^{T} \mathcal{L}_{\text{PPO}}\left(r_{\text{ppo},t}^{\text{rollout}}, A_t\right). \tag{9}$$

tis-srs-k3-corr-ratio. Token-level truncation followed by sequence-level rejection, using the correction ratio r_{corr} as the filtering signal:

$$\mathcal{L}_{\text{tis-srs-k3-corr}} = \mathbf{1}\left[\sum_{t=1}^{T} K3(r_{\text{corr},1:T}) \le \tau_{\text{seq}}\right] \sum_{t=1}^{T} \min\left(r_{\text{corr},t}, \tau_{\text{tok}}\right) \mathcal{L}_{\text{PPO}}\left(r_{\text{ppo},t}^{\text{train}}, A_t\right). \tag{10}$$

tis-srs-k1-corr-ratio. Token-level truncation followed by sequence-level rejection, using the correction ratio r_{corr} as the filtering signal:

$$\mathcal{L}_{\text{tis-srs-k1-corr}} = \mathbf{1}\left[\sum_{t=1}^{T} K1(r_{\text{corr},1:T}) \le \tau_{\text{seq}}\right] \sum_{t=1}^{T} \min\left(r_{\text{corr},t}, \tau_{\text{tok}}\right) \mathcal{L}_{\text{PPO}}\left(r_{\text{ppo},t}^{\text{train}}, A_t\right). \tag{11}$$

Threshold choice. The effectiveness of post-hoc algorithmic compensation depends on the token-level truncation and sequence-level rejection thresholds. We follow settings adopted in prior work (Qiu et al., 2026; Li et al., 2026). Specifically, we use $\tau_{\text{tok}} = 2$ for token-level truncated importance sampling and $\tau_{\text{seq}} = 0.001$ for sequence-level mismatch rejection. Our claim is not that these thresholds are suboptimal. Rather, without a zero-mismatch reference, threshold selection based only on observed training performance does not reveal how closely a correction recovers mismatch-free behavior. *ExactRL* provides such a reference and enables principled calibration of post-hoc corrections.