CPA: Efficient and Stable FP4 RL Training via Cross-Precision Alignment

Gu Gong[*‡] **Yining Wei**[*†] **Yuechen Tao**[§] **Tianyuan Wu**[†] **Peijie Dong**[‡]

Ruibo Fan[§] **Wenhu Hu**[§] **Yinghao Yu**[§] **Jiamang Wang**[§] **Wenbo Su**[§]

Guodong Yang[§] **Liping Zhang**[§] **Wei Wang**[†] **Xiaowen Chu**[†‡]

[†]Hong Kong University of Science and Technology
[‡]Hong Kong University of Science and Technology (Guangzhou)
[§]Alibaba Group

Abstract

Reinforcement learning (RL) for large language models (LLMs) is increasingly bottlenecked by rollout cost, making low-precision rollout appealing for acceleration. However, applying low-precision formats such as FP4 to RL remains unstable, even with quantization-aware training (QAT) enhancements. We identify that the root cause is that FP4's aggressive quantization amplifies small system-level numerical differences (i.e., training-inference stack mismatch) that are harmless at BF16 into large token-level log-probability errors, distorting importance weights and destabilizing optimization. To address this, we propose Cross-Precision Alignment (CPA), a simple regularizer that stabilizes low-precision RL by maintaining a BF16 master policy, executing rollouts in real FP4, and aligning the fake-quantized forward pass with a BF16 reference forward pass on sampled tokens by applying a low-variance KL penalty at each training step. In our evaluation, the resulting BF16 checkpoint matches a full BF16 RL baseline, and the overall pipeline delivers 8–19% end-to-end training speedup.

1 Introduction

Reinforcement learning (RL) post-training has become a central stage in improving large language models (LLMs), especially for reasoning-intensive tasks [19, 14]. In RL post-training, each iteration alternates between two phases: *rollout*, which generates responses under the current policy, and *policy update*, which improves the policy using the collected trajectories. For modern reasoning models, rollout is often the dominant cost, consuming approximately 70% of the total latency [11], because decoding is autoregressive and responses are long. This makes rollout the natural target for reducing end-to-end training time.

The increasing support of 4-bit floating point (FP4) execution by recent hardware and software [31, 32] makes low-precision rollout an attractive way to accelerate RL post-training. Prior to low-precision RL, recent work [6, 24, 41] has demonstrated that quantization can be effective for LLM inference and, more recently, even for pretraining [1, 4, 7]. RL post-training, however, is much more fragile than standard serving or pretraining because policy-gradient updates are driven by token-level log-

[*]Equal Contribution

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

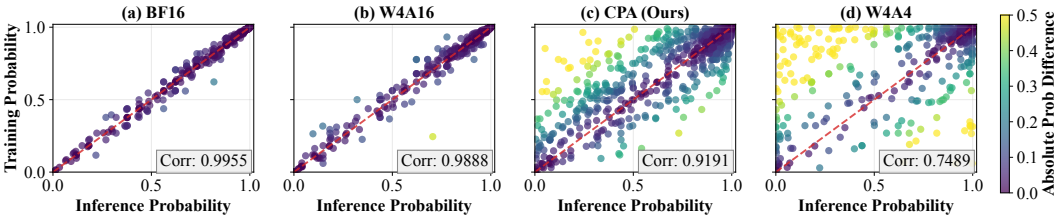


Figure 1: Training-inference mismatch severity at step 300 of Qwen3-1.7B (GRPO). The red line ($y = x$) represents an ideal case without numerical differences (mismatch). For BF16 and W4A16, the mismatch is slight and harmless. However, naive W4A4 *non-linearly amplifies* this otherwise benign numerical difference and destabilizes training (d). CPA mitigates this effect (c).

probability ratios [37, 35]: even small numerical errors in token log-probabilities can be amplified into large importance-weight deviations, directly corrupting gradient estimates.

Therefore, naively pushing FP4 into RL causes severe instability: fully low-precision RL training collapses quickly, and even the seemingly safer mixed-precision setting—FP4 rollout with higher-precision policy updates—remains unstable. Recent concurrent efforts on quantized RL, such as FlashRL [25], QuRL [22], QaRL [13], and related methods, address this instability via estimator- or update-level remedies. These state-of-the-art solutions suggest to execute rollout in real FP4 for acceleration, while keeping optimization in BF16 with quantization-aware training (QAT) [26, 8]. QAT simulates the FP4 quantization error by adding a fake quantization process during policy update so that the learned policy can be more robust under lower precision. This approach preserves stability well for conservative W4A16 (weights-only) rollout quantization [20, 36], but its efficiency gain is limited (see Figure 6). On the other hand, such corrections like FlashRL's truncated importance sampling (TIS) [25] are still insufficient in stabilizing the more aggressive W4A4 (both weights and activations in FP4) quantization, where its accuracy degrades sharply (see Figure 2).

Our diagnosis reveals that the root cause is not low precision alone, but the interplay of *cross-stack mismatch* and *FP4 amplification*: in a hybrid RL pipeline [38, 39], rollout (e.g., vLLM [21]) and training (e.g., FSDP [47]) run on different software stacks and produce slightly different token probabilities for the same weights due to system-level numerical differences (see Figure 1 (a) and (b)). As illustrated in Figure 1 (d), however, FP4 acts as a *nonlinear amplifier* under W4A4: it non-uniformly amplifies a small set of outlier tokens' perturbations to much larger activation and logit errors, turning an otherwise benign numerical gap into destructive probability errors that destabilize policy-gradient updates.

This observation also suggests the right intervention. Since the problematic discrepancy is partly induced by stack-level numerics, it cannot be removed reliably by trying to match the rollout and training implementations more closely. Instead, we *align the FP4 quantized model's behavior to its higher-precision version* (see Figure 1 (c) for its effectiveness). Concretely, we keep a BF16 master policy, execute rollout in real FP4, and during training run two forwards on the sampled trajectories: a fake-quantized and a BF16 reference. We then add CPA, a token alignment loss that pulls the fake-quantized policy toward the BF16 reference on the tokens that directly drive the RL objective.

A key property of this design is that the training-side alignment target is not arbitrary: it penalizes the same quantization error source that will later appear as rollout-training mismatch. Empirically, the BF16-to-fake-quant discrepancy and the cross-stack mismatch are strongly correlated on the same tokens (Section 3.3), and strengthening the training-side alignment correspondingly improves the statistics of cross-stack mismatch. Thus, although CPA is applied entirely within the training stack, it reduces the rollout-side mismatch that destabilizes low-precision RL.

This design preserves the main efficiency benefit of FP4: rollout runs in real FP4, reducing the cost of the dominant phase of RL post-training. The final model remains a standard BF16 checkpoint, but one that is also more robust to subsequent FP4 quantization.

In summary, this paper makes the following contributions:

- We identify a practical failure behavior in low-precision RL post-training: small rollout-training numerical mismatches become destructive under FP4 because it amplifies them into large token-

level probability errors. This mismatch-amplification effect, rather than low precision alone, destabilizes RL.
- We propose CPA, a simple training-side cross-precision alignment method that combines real-FP4 rollout, fake-quantized training, and a BF16 reference forward. CPA aligns the fake-quantized policy with the BF16 policy on sampled tokens, thereby reducing the quantization sensitivity that gives rise to cross-stack mismatch.
- We show that CPA stabilizes FP4 RL post-training while preserving efficiency benefits and model quality: the BF16 checkpoint matches the BF16 RL baseline and achieves 8–19% end-to-end pipeline speedup.

2 Preliminary

2.1 Group Relative Policy Optimization

Group Relative Policy Optimization (GRPO) [37] and its variants, such as DAPO [45], are widely used algorithms for RL post-training of large language models. Let π_θ denote the policy parameterized by θ. For each query x, the rollout policy $\pi_{\theta_{\text{old}}}$ samples a group of N responses, denoted by $\{y_i\}_{i=1}^{N}$. A reward model or verifier then assigns a scalar reward $r(x, y_i)$ to each response.

GRPO computes advantages by normalizing rewards within each group: $\hat{A}_i = (r(x, y_i) - \mu)/\sigma$, where μ and σ are the mean and standard deviation of the rewards associated with the same query x. Let $\rho_{i,t} = \pi_\theta(y_{i,t} \mid x)/\pi_{\theta_{\text{old}}}(y_{i,t} \mid x)$ denote the per-token likelihood ratio, where t is the token index, Let ϵ be the clipping hyperparameter, the GRPO objective is

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\Big(\rho_{i,t}\hat{A}_i,\ \text{clip}(\rho_{i,t}, 1-\epsilon, 1+\epsilon)\hat{A}_i\Big)\right]. \tag{1}$$

DAPO follows the same overall framework, while introducing refinements such as decoupled clipping and dynamic sampling to improve stability and sample efficiency. In this paper, we adopt GRPO as the main objective, and our analysis also extends to DAPO, as evaluated in Section 4.

2.2 FP4 Quantization and Quantization-Aware Training (QAT)

FP4 Quantization. Quantization is appealing for RL post-training for its reduced memory usage and computation like GEMMs. In particular, FP4 has recently become a practical choice, as Blackwell-class GPUs provide native support for FP4 formats such as NVFP4 [31] and MXFP4 [32]. FP4 uses an E2M1 floating-point format with additional scaling factors to improve dynamic range and precision.

Although mitigated by the NVFP4's scaling mechanisms, the efficiency gain still comes at the cost of limited precision. We first formally define the quantization error. Let $Q(\cdot)$ denote an FP4 quantizer. For a tensor z, its quantized value can be written as

$$\hat{z} = Q(z) = \left[\frac{z/s}{\max|z/s|}\cdot\alpha\right]_{\text{FP4}}, \tag{2}$$

where s is the scaling factor, $[\cdot]_{\text{FP4}}$ denotes casting to FP4, and $\alpha = 6$ is the maximum representable value for E2M1 FP4. In NVFP4, s is the product of the local block factor s_l and a global factor s_g. Let $\hat{W} = Q(W)$ denote the quantized weights. The corresponding error is

$$\Delta_W = W - \hat{W}. \tag{3}$$

More importantly, quantization perturbs the layer output. For an input activation x, bias b, and activation function $\phi(\cdot)$, the full-precision output is

$$h = \phi(Wx + b), \tag{4}$$

whereas a full W4A4-quantized (i.e., 4-bit for weights, 4-bit for activations) forward computes

$$\hat{h}_{\text{W4A4}} = \phi(\hat{W}\hat{x} + \hat{b}), \tag{5}$$

We therefore define the induced activation (or layer-output) error as

$$\Delta_{h,\mathrm{W4A4}} = h - \hat{h}_{\mathrm{W4A4}}. \tag{6}$$

For the W4A16 setting that keeps activations in BF16 (i.e., $\hat{x} = x$), quantizing the weights still introduces activation error $\Delta_{h,\mathrm{W4A16}} = \phi(\hat{W}x + \hat{b}) - h$.

Quantization-Aware Training (QAT). QAT [26, 8] is a standard way to mitigate this error, which inserts quantize-dequantize operators in the forward pass to simulate FP4 rounding, while keeping the master weights and optimization states in higher precision. It is important to distinguish this from real FP4 execution: a *real FP4 forward* dispatches linear layers to native FP4 GEMM kernels and performs the computation on low-precision operands directly, whereas a *fake-quantized FP4 forward* emulates FP4 numerics inside a higher-precision kernel by applying quantize-then-dequantize before the matrix multiplication. We denote the resulting policy by π_θ^{FQ}.

2.3 Hybrid-Engine RL and Mismatch

Modern RL pipelines for LLMs typically adopt a hybrid-engine design to improve efficiency. During rollout, an inference engine such as vLLM [21] generates trajectories using a *real quantized* policy, denoted by $\hat{\pi}_{\theta_{\mathrm{old}}}$. During training, a separate engine such as FSDP [47] recomputes log-probabilities and evaluates gradients using a *fake-quantized policy*, denoted by π_θ^{FQ}.

Although the two policies are intended to represent the same model, this setup introduces a rollout-training mismatch [43, 44]. Even with identical weights, the inference and training stacks can produce slightly different token probabilities because they rely on different kernel implementations and numerical execution paths, such as fused versus unfused GEMMs and different accumulation precisions [13]. As a result, tokens are sampled from $\hat{\pi}_{\theta_{\mathrm{old}}}$ on the rollout side, while the GRPO importance ratio is computed entirely on the training side using $\rho_t = \pi_\theta^{\mathrm{FQ}}(t)/\pi_{\theta_{\mathrm{old}}}^{\mathrm{FQ}}(t)$.

This discrepancy makes the update effectively off-policy. Following recent work [25], we correct it with truncated importance sampling (TIS), which augments the policy gradient with a clipped correction factor $\min(c, \exp(\Delta_t^{\mathrm{stack}}))$, where

$$\Delta_t^{\mathrm{stack}} \triangleq \log \pi_\theta^{\mathrm{FQ}}(t) - \log \hat{\pi}_\theta(t) \tag{7}$$

denotes the per-token log-probability discrepancy between the training and rollout stacks. This hybrid vLLM-FSDP pipeline with TIS correction is the **baseline setup** used in this paper.

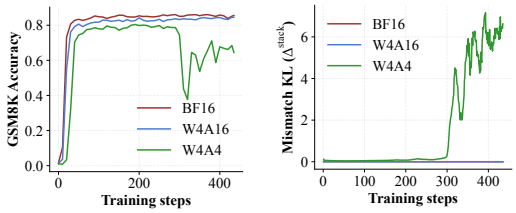
3 Methodology

Applying FP4 to RL post-training remains unstable even with TIS correction. We first diagnose the failure mode and show that W4A4 quantization *amplifies small rollout-training numerical mismatches into large token-level errors* (Section 3.1). We then introduce CPA, a regularizer that suppresses this amplification at training time (Section 3.2), and finally explain why this training-side alignment also reduces the cross-stack mismatch between real-FP4 rollout and fake-quantized training (Section 3.3).

3.1 Diagnosis: W4A4 Quantization Acts as a Nonlinear Amplifier

Failure behavior. Figure 2 shows the behavior of the baseline hybrid RL pipeline described in Section 2.3: rollout uses real W4A4 with vLLM, training uses BF16 FSDP with fake quantization, and TIS corrects the resulting stack mismatch. This setup appears stable for roughly the first 300 steps, but then becomes unstable: the stack discrepancy $\Delta_t^{\mathrm{stack}}$ rises sharply, and GSM8K accuracy drops accordingly at the same time. In contrast, otherwise identical BF16 and W4A16 runs train smoothly to convergence. This comparison isolates the main destabilizing factor in our setting is the *FP4-quantized activations*, rather than the whole low-precision RL pipeline.

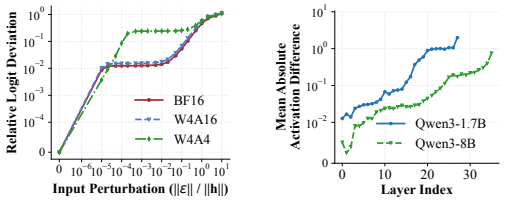
Root cause: FP4 amplifies otherwise benign system noise. The perturbation originates from the hybrid rollout-training stack. Even with identical weights, engines such as vLLM and FSDP produce small numerical differences because they use different execution paths, such as fused versus unfused



(a) Training dynamics of BF16/W4A16/W4A4.

(b) Mismatch (Δ^{stack}) of BF16/W4A16/W4A4.

Figure 2: Failure behavior of applying W4A4 into RL post-training (with QAT and TIS enabled).



(a) Output deviation vs. input perturbation.

(b) Real RL activations fall in the sensitive region.

Figure 3: FP4 can amplify small perturbations; Activation falls in the amplification range.

GEMMs and different accumulations [16, 46]. In higher precision, these differences are usually harmless, where BF16 and W4A16 remain stable under the same hybrid stack (Figure 2).

Under W4A4, however, the same perturbation is no longer benign. To measure this effect, we take a real activation tensor h from Qwen3-1.7B, add a random perturbation ϵ with relative magnitude $\|\epsilon\|/\|h\|$ swept from 0 to 10^1, and track the resulting relative logit deviation through the rest of the network. Figure 3a shows that for perturbations in the range 10^{-4} to 10^0, W4A4 *non-linearly* amplifies the output error to roughly 10^0, whereas W4A16 and BF16 keep the error near 10^{-2}. Figure 3b further shows that the activation discrepancies observed in real Qwen3-1.7B and 8B training with hybrid stacks *fall within this sensitive regime* and will be inevitably amplified.

The reason of amplification is that FP4 is a *nonlinear quantizer*. It can represent only a small set of non-uniform values, e.g., $\pm\{0, 0.5, 1, 1.5, 2, 3, 4, 6\}$. As the magnitude grows, the gap between adjacent representable values also grows. Therefore, if an activation lies between two large values, a small perturbation can change its quantized result by a relatively large amount. For example, an activation of 4.95 in rollout stack (e.g., vLLM) is rounded to 4. A perturbation of only 0.1 introduced by the training stack changes it to 5.05, which is rounded to 6. Thus, a tiny input perturbation can produce a much larger mismatch after FP4 quantization.

This amplification non-uniformly creates a small set of *outlier tokens* with large mismatch values. In our runs, this effect is visible directly through TIS: the clip-trigger rate $\Pr(\exp(\Delta_t^{\text{stack}}) > c)$ rises from about 0.01% under W4A16 to 3.77% under W4A4, a roughly $400\times$ increase. Thus, even with TIS, these outlier tokens appear far more often under W4A4 and can disproportionately affect the batch gradient. Their influence accumulates over training and destabilizes the training curve.

Implication for method design. This amplification is especially harmful in RL, where the algorithm forms token-level ratios and correction factors from log-probabilities, so a few large-error tokens can dominate the update. More importantly, the effect is structural: even if the training and rollout stacks are matched more carefully, any residual numerical delta can still be amplified by W4A4 in the same way. This suggests that *the right intervention is not to chase exact stack-level equivalence, but to constrain the policy so that FP4 quantization only induces limited behavioral shifts on the RL objective*. We design CPA to enforce this property during training.

3.2 CPA: Training-Side Cross-Precision Policy Alignment

The hybrid rollout-training pipeline and the amplified quantization error introduce a practical difficulty for optimization. A natural target would be the rollout-side discrepancy between the FP4 policy used by the inference engine (i.e., $\hat{\pi}_{\theta_{\text{old}}}$) and the policy used on the training side (i.e., π_θ^{FQ}). However, this discrepancy originates from the unpredictable cross-stack system noises (and amplified by the low-precision), which are fully stochastic and *not directly learnable* by the actor.

To address this, we optimize a differentiable surrogate on the training side. At each training step, we run two forwards of the same model weights θ on the rollout buffer. The first is a BF16 forward, denoted by π_θ^{BF}, with fake quantization disabled. It serves as a frozen teacher that represents the *current model behavior without FP4 perturbation*. The second is the fake-quantized forward π_θ^{FQ},

serving as a *trainable student*. For each sampled token t, we define the training-side discrepancy

$$\Delta_t^{\text{CPA}} \triangleq \log \pi_\theta^{\text{BF}}(t) - \log \pi_\theta^{\text{FQ}}(t). \tag{8}$$

This term measures how much π_θ^{FQ} changes the token-level behavior of the current policy π_θ^{BF}.

We choose to regularize the policy at the token-logprob level rather than in weight space. A direct penalty such as $\|Q(\theta) - \theta\|^2$ only measures parameter distortion [6, 24, 41], while RL updates are driven by token probabilities on sampled trajectories. In contrast, Δ_t^{CPA} directly captures the behavior shift induced by quantization on the tokens that participate in policy optimization.

Based on this discrepancy, we define the CPA loss using a non-negative KL-form penalty, following the form used in GRPO [37]:

$$\mathcal{L}_{\text{CPA}} = \frac{1}{M} \sum_{t=1}^{M} \left(\exp\left(-\Delta_t^{\text{CPA}}\right) - 1 + \Delta_t^{\text{CPA}} \right), \tag{9}$$

where M is the number of sampled tokens in the batch. This objective is always non-negative and penalizes deviations between the BF16 teacher and the fake-quantized student. It therefore acts as a *bridge* term that pulls the trainable QAT policy toward the corresponding BF16 behavior.

Unlike GRPO Loss, CPA is averaged uniformly over sampled tokens rather than weighted by advantages. This design keeps CPA as an auxiliary policy-preservation regularizer, instead of coupling it to reward magnitude and overemphasizing a small subset of high-advantage tokens.

Combining the CPA loss into GRPO training pipeline, our final training objective is

$$\mathcal{L} = \mathcal{L}_{\text{GRPO+TIS}} + \lambda_{\text{CPA}} \, \mathcal{L}_{\text{CPA}}. \tag{10}$$

Here, $\mathcal{L}_{\text{GRPO+TIS}}$ denotes the original GRPO loss (Section 2.1) with TIS correction (Section 2.3), λ_{CPA} controls the strength of the auxiliary regularizer.

Distinction from a static reference KL. Although $\mathcal{L}_{\text{CPA}}$ resembles the reference-KL term in standard RLHF [33, 37], its two branches π_θ^{BF} and π_θ^{FQ} share the same trainable θ of the *latest* policy model, rather than anchoring the policy to a frozen pre-RL model.

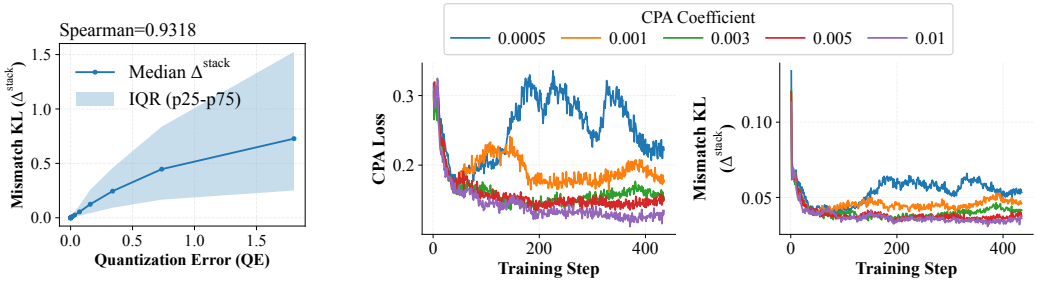
3.3 Why Training-Side Alignment Reduces Cross-Stack Mismatch

CPA penalizes the training-side discrepancy Δ^{CPA}, whereas the failure in Section 3.1 is caused by the cross-stack discrepancy Δ^{stack}. We show that penalizing Δ^{CPA} also suppresses Δ^{stack}.

Correlation between the two discrepancies. We first measure both Δ^{stack} and Δ^{CPA} on the same set of 250k tokens from a fixed checkpoint: Δ_t^{CPA} is computed as the BF16–fake-quant discrepancy inside FSDP, while Δ_t^{stack} is computed as the fake-quant–real-FP4 discrepancy across FSDP and vLLM. On sampled tokens, the two are strongly monotonic, with Spearman $\rho = 0.9318$ (Figure 4a). In other words, tokens that exhibit large BF16-to-fake-quant deviation during training are also the tokens that exhibit large training-to-rollout mismatch across stacks.

This correlation is expected because both discrepancies reflect the same underlying property of the policy: its sensitivity to FP4 quantization. Δ^{CPA} measures how much fake quantization changes the current BF16 policy within the training stack, while Δ^{stack} measures how that same quantized behavior diverges when executed through the real FP4 rollout stack. Since both pass through the same FP4 binning structure identified in Section 3.1, reducing the policy's quantization sensitivity on the training side also suppresses the mismatch that appears across stacks.

Effect and trade-off of CPA strength. We observe the same correlation and trend when varying the strength of CPA. As λ_{CPA} increases, the CPA loss decreases, and the statistics of Δ_t^{stack} improve accordingly. This trend shows in Figure 4b that training-side cross-precision alignment translates into better rollout-training consistency in practice. At the same time, the coefficient cannot be increased indefinitely: overly strong alignment over-constrains the policy and slows RL improvement. We will further demonstrate this trade-off and coefficient selection in Section 4.



(a) Correlation of QE and Δ^{stack}.

(b) The CPA Loss and Δ^{stack} under different CPA strength.

Figure 4: (a) QE is monotonic to mismatch KL. (b) Impact of the regularization strength (λ_{CPA}).

4 Experiments

We evaluate CPA along three research questions. **Q1 (Training quality).** Can CPA stabilize FP4 RL post-training while matching the downstream accuracy of an unquantized BF16 baseline (Section 4.2)? **Q2 (Efficiency).** How much end-to-end speedup does CPA achieve through low-precision rollout (Section 4.3)? **Q3 (Sensitivity).** How sensitive is CPA to the choice and strength of the regularization term (Section 4.4)?

4.1 Experimental Setup

Tasks and models. We evaluate CPA across two representative RL post-training regimes, covering both the GRPO [37] and DAPO [45] algorithms as well as two backbone scales. For GRPO, we train **Qwen3-1.7B-Base** [42] on GSM8K [5] and report zero-shot accuracy on GSM8K, MATH-500 [23], and SVAMP [34]. For DAPO, we train **Qwen3-8B-Base** [42] on DAPO-Math-17k [45], and evaluate on AIME24 [29], AIME25 [30], AMC23 [28], and MATH-500 [23]. We choose these settings because prior work [13, 22] reports substantial instability for low-precision RL in precisely these regimes, making them a stringent test of CPA.

Baselines. We compare CPA against five baselines. **Base** is the pre-RL backbone and serves as a no-post-training reference. **BF16** is the standard full-precision RL baseline and provides the main quality target. **Naive W4A4** applies QAT-based W4A4 RL without CPA, isolating the effect of CPA under the same low-precision setting. We also compare against two recent low-precision RL methods, **QaRL** [13] and **QuRL** [22].

Implementation details. All low-precision methods, including Naive W4A4, QaRL, QuRL, and CPA, use the same hybrid pipeline: rollout is executed in real NVFP4 on vLLM, while training is performed with FSDP using BF16 master weights and a fake-quantized forward that simulates FP4 error in the forward pass and uses straight-through [2] gradients in the backward pass. QaRL and QuRL are reproduced in our codebase, with their original INT4 W4A16 settings adapted to the FP4 W4A4 setting studied here. For CPA, we use the low-variance KL objective (Equation (9)) with default coefficient $\lambda_{\text{CPA}} = 1\text{e-}3$. More detailed experimental settings are provided in Appendix B.

Evaluation. We evaluate the BF16 master weights from the final checkpoint. For GSM8K, MATH-500, and SVAMP, we generate 8 samples per prompt; for AIME and AMC, we generate 32 samples per prompt. We then report the average accuracy over all generated samples across the evaluation set.

4.2 Main Results

Figure 5 shows that CPA produces markedly more stable training dynamics. Its step-to-accuracy trajectory remains close to the unquantized BF16 baseline, and its mismatch KL stays consistently lower and more stable than the other low-precision methods. In contrast, Naive W4A4 and QuRL exhibit clear late-stage degradation, with collapsing accuracy trends and larger mismatch spikes.

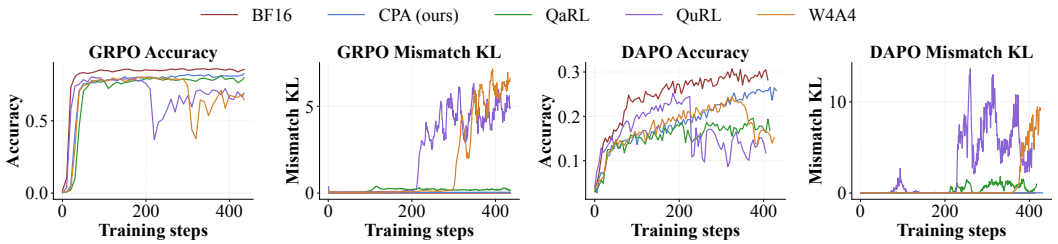


Figure 5: Training dynamics in the main-result settings. Across both GRPO and DAPO recipes, CPA consistently keeps mismatch KL low, whereas other low-precision baselines show large spikes and oscillations, indicating less stable optimization. The visible accuracy gap between CPA and BF16 during training is expected because these curves are evaluated in W4A4 precision; BF16 serves as a high-precision upper-bound reference rather than a same-precision runtime trajectory.

Table 1: Main results under GRPO (Qwen3-1.7B-Base) and DAPO (Qwen3-8B-Base). All numbers are accuracy (%) averaged over multiple samples on the validation split of each benchmark, evaluated on the BF16 master weights at the final RL step. **Bold** marks the best low-precision result per column.

Method	GRPO (Qwen3-1.7B-Base)				DAPO (Qwen3-8B-Base)				
	GSM8K	MATH-500	SVAMP	Avg.	AIME24	AIME25	AMC	MATH-500	Avg.
Qwen3-Base	50.3	51.6	71.3	57.7	3.2	2.2	40.8	66.4	28.2
BF16	85.4	60.6	88.7	78.2	30.3	24.3	58.1	77.1	47.5
Naive W4A4	70.4	60.0	73.7	68.0	18.8	18.5	47.0	72.4	39.1
QuRL	78.2	50.4	85.0	71.2	13.9	13.3	30.8	39.2	24.3
QaRL	83.2	58.0	87.7	76.3	20.9	18.0	54.2	74.4	41.8
CPA (ours)	**86.1**	**60.2**	**88.0**	**78.1**	**30.3**	**20.9**	**59.8**	**77.6**	**47.1**

To assess the final model quality, we evaluate each baseline's last BF16 checkpoint and summarize the downstream results in Table 1. Across both GRPO and DAPO, CPA remains close to BF16 on average, with only about 0.1% and 0.4% accuracy drop, accordingly. On several benchmarks, it even slightly exceeds BF16, including GSM8K (85.4% vs. 86.1%) with GRPO and AMC (58.1% vs. 59.8%) and MATH-500 (77.1% vs. 77.6%) with DAPO. In some harder benchmarks, slight accuracy gaps remain. For example, CPA reaches 20.9% on AIME25 compared with 24.3% for BF16, but it still substantially outperforms the other low-precision methods, such as QuRL at only 13.3%.

Across all evaluated tasks, CPA is the strongest low-precision method. It consistently improves over Naive W4A4, and the margin over QuRL reaches as high as 38.4 percentage points on MATH-500 in the DAPO setting. It also outperforms QaRL across the reported benchmarks. Overall, CPA achieves stable training dynamics while preserving downstream accuracy close to BF16, clearly surpassing prior low-precision RL methods.

4.3 Efficiency

Figure 6 reports per-step training time and its breakdown on our two main experimental settings (GRPO with Qwen3-1.7B, DAPO with Qwen3-8B). To assess scalability, we also include a larger GRPO setting with a Qwen3 32B backbone. We compare CPA against BF16 and W4A16 baselines. Across all three settings, the overall speedup comes primarily from faster FP4 rollout: CPA reduces rollout time relative to BF16 by 21.4%, 31.3%, and 51.0% at 1.7B, 8B, and 32B, correspondingly. The additional BF16 reference forward introduces only modest training overhead, increasing training time by 9.1%, 16.0%, and 15.0% at 1.7B, 8B, and 32B, correspondingly. As a result, CPA reduces end-to-end per-step training time by 8.3%, 19.3%, and 18.1% relative to BF16, respectively.

The gain is smaller at 1.7B because the model is too small to fully utilize GPU compute even under BF16 rollout, leaving limited room for FP4 to further accelerate generation. As model scale increases, rollout becomes a larger bottleneck and FP4 delivers substantially larger savings.

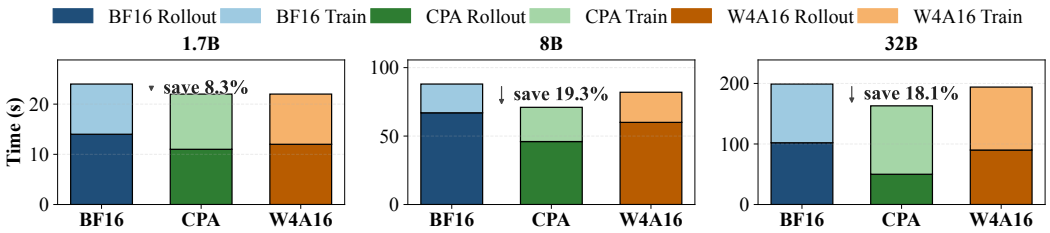


Figure 6: End-to-end per-step time and breakdown across 1.7B (GRPO), 8B (DAPO), and 32B (GRPO). We compare CPA against BF16 and W4A16 baselines.

Table 2: Loss-form mean±std across the coefficient sweep ($\lambda_{\text{CPA}} \in \{0.001, 0.003, 0.005, 0.01\}$).

Loss form	GRPO Avg±Std	DAPO Avg±Std
`low_var_kl` (ours)	**77.6±0.6**	**45.8±1.2**
`mse`	76.9±0.6	**45.8±1.0**
`mae`	70.3±9.7	45.6±1.5

4.4 Ablation Study

To evaluate the sensitivity of regularization form and strength, we ablate both the form of the per-token penalty applied to the training-side discrepancy Δ_t^{CPA} and the regularization strength λ_{CPA}. Specifically, we compare three penalty forms: the `low_var_kl` objective used in CPA (Equation (9)), a mean squared penalty $\sum_{t=1}^{M}(\Delta_t^{\text{CPA}})^2/M$ (`mse`), and a mean absolute penalty $\sum_{t=1}^{M}|\Delta_t^{\text{CPA}}|/M$ (`mae`). For each variant, we sweep $\lambda_{\text{CPA}} \in \{0.001, 0.003, 0.005, 0.01\}$.

Table 2 summarizes the average (Avg) and standard deviation (Std) over this sweep. We can observe that both `low_var_kl` and `mse` remain consistently stable across all four coefficients, with similar mean scores and stds are no larger than 1.2 in either GRPO or DAPO. In contrast, `mae` is noticeably less reliable: while it remains competitive on DAPO, it exhibits much larger variance on GRPO (std 9.7). A plausible reason is that the linear penalty provides a constant-magnitude gradient, regardless of the size of Δ_t^{CPA}, and therefore corrects large outliers less aggressively.

5 Related Work

RL training acceleration. Since rollout dominates RL post-training cost, a growing line of system-level work targets GPU underutilization through long-tail rollout scheduling [11, 48, 17], asynchronous rollout–training decoupling [10, 15, 27], hybrid-engine execution [38, 39], and cross-cluster co-scheduling for disaggregated training [40, 12]. These methods reduce wall-clock cost through execution scheduling and resource placement, leaving the per-token computation cost unchanged. Our work is orthogonal: by stabilizing W4A4 RL, we reduce the per-token computation cost via FP4 execution and the resulting speedup is expected to compound with these optimizations.

Quantization. Standard LLM quantization spans PTQ [9, 24], QAT [26, 3], and fully quantized training [4, 1, 7]; Several recent methods combine quantization with RL: FlashRL [25] introduces Truncated Importance Sampling (TIS) to correct quantized-rollout mismatch, QuRL [22] stabilizes RL via decoupled PPO with adaptive clipping, QaRL [13] aligns the training-side forward to the quantized rollout, and QeRL [18] treats NVFP4 noise as exploration in LoRA. All these previous methods operate at W8A8 or W4A16, forgoing the native FP4 GEMM path on the latest generation of GPUs. Pushing to W4A4 breaks importance-sampling-based correction, as FP4 nonlinearly amplifies stack-level mismatches into heavy-tailed token-level errors. We address this with a direct penalty on the Δ_t^{CPA} at training time, complementary to TIS rather than a replacement.

6 Conclusions

We present CPA, a simple and effective regularizer for stabilizing W4A4 FP4 RL post-training in the hybrid rollout-training stack. We trace the failure mode of W4A4 RL to FP4's role as a nonlinear amplifier: it magnifies otherwise benign cross-stack numerical differences into large token-level logprob errors, which then distort importance weights and destabilize policy optimization. CPA addresses this at the source by aligning the fake-quantized forward with a BF16 reference forward at the token-logprob level during training. The resulting pipeline retains real FP4 rollout for efficiency, matches the BF16 RL baseline in end-task accuracy, and delivers 8–19% end-to-end per-step speedup across 1.7B, 8B, and 32B backbones.

Limitations.

- **Training-side overhead.** The additional BF16 forward adds about 5% training overhead relative to naive W4A4, leaving room for further training-side optimization.
- **Coefficient tuning.** λ_{CPA} trades off alignment against the policy-gradient signal. Although training remains stable across our sweep, the optimal coefficient may vary with model scale, task, or rollout length.
- **Evaluation scope.** Our training-quality evaluation covers Qwen3 math reasoning under GRPO/DAPO with vLLM–FSDP and NVFP4. Generalization to other model families, domains, stacks, or FP4 formats, and training stability beyond 8K response tokens, remain unverified.
- **Baseline tuning.** We retain the reported hyperparameters of QaRL and QuRL when adapting them to FP4 W4A4, without additional per-baseline sweeps. Further tuning for this precision setting may improve their performance.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for their constructive feedback, which helped improve this paper. This work was supported in part by the Alibaba Innovative Research (AIR) Grant and Hong Kong CRF grants C7004-22G and C6015-23G. The corresponding author is Xiaowen Chu. The authors have no additional competing interests to declare.

References

[1] Felix Abecassis, Anjulie Agrusa, Dong Ahn, Jonah Alben, Stefania Alborghetti, Michael Andersch, Sivakumar Arayandi, Alexis Bjorlin, Aaron Blakeman, Evan Briones, et al. Pretraining large language models with nvfp4. *arXiv preprint arXiv:2509.25149*, 2025.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[3] Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. Efficientqat: Efficient quantization-aware training for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10081–10100, 2025.

[4] Brian Chmiel, Maxim Fishman, Ron Banner, and Daniel Soudry. Fp4 all the way: Fully quantized training of llms. *arXiv preprint arXiv:2505.19115*, 2025.

[5] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[6] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35:30318–30332, 2022.

[7] Peijie Dong, Ruibo Fan, Yuechen Tao, Di Mou, Wenhu Hu, Zhenheng Tang, Yinghao Yu, Jiamang Wang, Wenbo Su, Guodong Yang, et al. Dissecting outlier dynamics in llm nvfp4 pretraining. *arXiv preprint arXiv:2602.02047*, 2026.

[8] Dayou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, Xiaowen Chu, and Ningyi Xu. Bitdistiller: Unleashing the potential of sub-4-bit llms via self-distillation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 102–116, 2024.

[9] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[10] 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*, 2025.

[11] Wei Gao, Yuheng Zhao, Dakai An, Tianyuan Wu, Lunxi Cao, Shaopan Xiong, Ju Huang, Weixun Wang, Siran Yang, Wenbo Su, et al. Rollpacker: Mitigating long-tail rollouts for fast, synchronous rl post-training. *arXiv preprint arXiv:2509.21009*, 2025.

[12] Wei Gao, Yuheng Zhao, Tianyuan Wu, Shaopan Xiong, Weixun Wang, Dakai An, Lunxi Cao, Dilxat Muhtar, Zichen Liu, Haizhou Zhao, et al. Rollart: Scaling agentic rl training via disaggregated infrastructure. *arXiv preprint arXiv:2512.22560*, 2025.

[13] Hao Gu, Hao Wang, Jiacheng Liu, Lujun Li, Qiyuan Zhu, Bei Liu, Binxing Xu, Lei Wang, Xintong Yang, Sida Lin, et al. Qarl: Rollout-aligned quantization-aware rl for fast and stable training under training–inference mismatch. *arXiv preprint arXiv:2604.07853*, 2026.

[14] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[15] Zhenyu Han, Ansheng You, Haibo Wang, Kui Luo, Guang Yang, Wenqi Shi, Menglong Chen, Sicheng Zhang, Zeshun Lan, Chunshi Deng, et al. Asyncflow: An asynchronous streaming rl framework for efficient llm post-training. *arXiv preprint arXiv:2507.01663*, 2025.

[16] Horace He and Thinking Machines Lab. Defeating nondeterminism in LLM inference. `https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/`, September 2025. Thinking Machines Lab: Connectionism.

[17] Jingkai He, Tianjian Li, Erhu Feng, Dong Du, Qian Liu, Tao Liu, Yubin Xia, and Haibo Chen. History rhymes: Accelerating llm reinforcement learning with rhymerl. *arXiv preprint arXiv:2508.18588*, 2025.

[18] Wei Huang, Yi Ge, Shuai Yang, Yicheng Xiao, Huizi Mao, Yujun Lin, Hanrong Ye, Sifei Liu, Ka Chun Cheung, Hongxu Yin, et al. Qerl: Beyond efficiency–quantization-enhanced reinforcement learning for llms. *arXiv preprint arXiv:2510.11696*, 2025.

[19] 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.

[20] Kimi Team, Moonshot AI. Kimi k2 thinking. `https://moonshotai.github.io/Kimi-K2/thinking.html`, 2025. Technical report; native INT4 W4A16 QAT.

[21] 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*, pages 611–626, 2023.

[22] Yuhang Li, Reena Elangovan, Xin Dong, Priyadarshini Panda, and Brucek Khailany. Qurl: Efficient reinforcement learning with quantized rollout. *arXiv preprint arXiv:2602.13953*, 2026.

[23] 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.

[24] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100, 2024.

[25] Liyuan Liu, Feng Yao, Dinghuai Zhang, Chengyu Dong, Jingbo Shang, and Jianfeng Gao. Flashrl: 8bit rollouts, full power rl, August 2025.

[26] Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 467–484, 2024.

[27] Han Lu, Zichen Liu, Shaopan Xiong, Yancheng He, Wei Gao, Yanan Wu, Weixun Wang, Jiashun Liu, Yang Li, Haizhou Zhao, et al. Part ii: Roll flash–accelerating rlvr and agentic training with asynchrony. *arXiv preprint arXiv:2510.11345*, 2025.

[28] Mathematical Association of America (MAA). AMC 2023. `https://artofproblemsolving.com/wiki/index.php/AMC_Problems_and_Solutions`, 2023. American Mathematics Competitions, 2023.

[29] Mathematical Association of America (MAA). AIME 2024. `https://artofproblemsolving.com/wiki/index.php/2024_AIME_I`, 2024. American Invitational Mathematics Examination, 2024.

[30] Mathematical Association of America (MAA). AIME 2025. `https://artofproblemsolving.com/wiki/index.php/2025_AIME_I`, 2025. American Invitational Mathematics Examination, 2025.

[31] NVIDIA. NVFP4: A new 4-bit floating-point format for efficient LLM training and inference. NVIDIA Developer Blog, 2024. Accessed: 2026-04.

[32] Open Compute Project. OCP microscaling formats (MX) specification, version 1.0. Technical report, Open Compute Project Foundation, 2023.

[33] 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.

[34] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 2080–2094, 2021.

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

[36] SGLang RL Team and InfiXAI Team and Ant Group Asystem & AQ Infra Team and slime Team and RadixArk Team. Squeezing 1tb model rollout into a single h200: Int4 qat rl end-to-end practice. `https://lmsys.org/blog/2026-01-26-int4-qat/`, January 2026. Open-source reproduction of Kimi K2's INT4 QAT for large-scale RL.

[37] 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.

[38] 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*, pages 1279–1297, 2025.

[39] Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, et al. Reinforcement learning for reasoning in large language models with one training example. *arXiv preprint arXiv:2504.20571*, 2025.

[40] Tianyuan Wu, Lunxi Cao, Yining Wei, Wei Gao, Yuheng Zhao, Dakai An, Shaopan Xiong, Zhiqiang Lv, Ju Huang, Siran Yang, et al. Rollmux: Phase-level multiplexing for disaggregated rl post-training. *arXiv preprint arXiv:2512.11306*, 2025.

[41] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pages 38087–38099. PMLR, 2023.

[42] 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.

[43] 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.

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

[45] 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.

[46] 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. *arXiv preprint arXiv:2511.17826*, 2025.

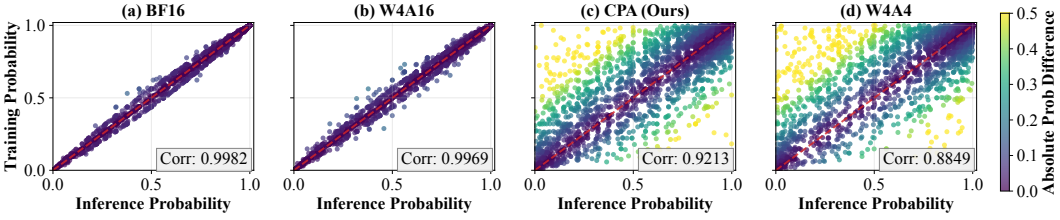
[47] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

[48] Yinmin Zhong, Zili Zhang, Xiaoniu Song, Hanpeng Hu, Chao Jin, Bingyang Wu, Nuo Chen, Yukun Chen, Yu Zhou, Changyi Wan, et al. Streamrl: Scalable, heterogeneous, and elastic rl for llms with disaggregated stream generation. *arXiv preprint arXiv:2504.15930*, 2025.

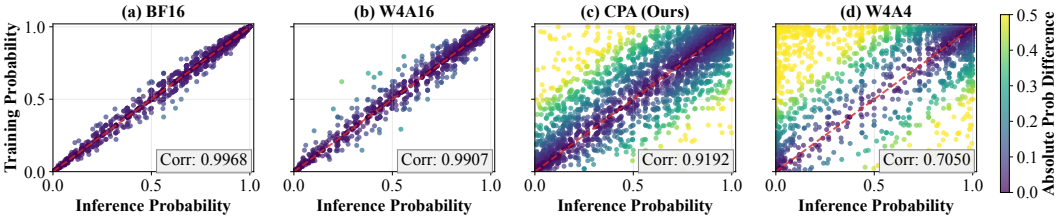
A Additional Analysis of Training–Inference Mismatch

A.1 Mismatch Severity During Training

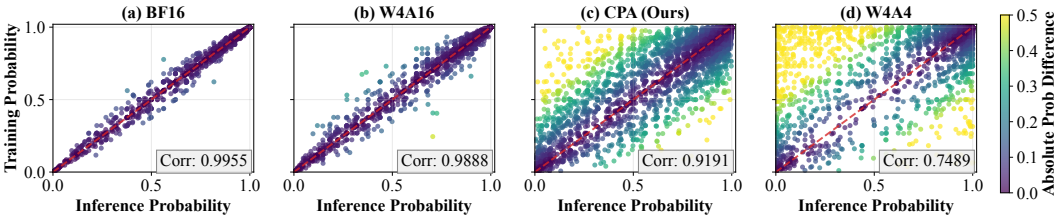
Figure 7 extends Figure 1 to steps 100–400 of Qwen3-1.7B training with GRPO. Mismatch grows under naive W4A4, while CPA maintains closer rollout–training agreement.



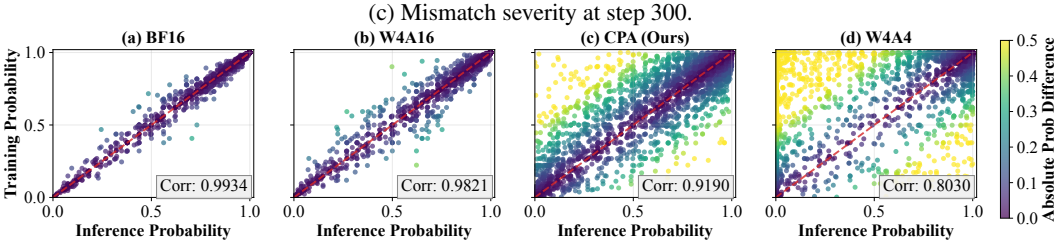
(a) Mismatch severity at step 100.



(b) Mismatch severity at step 200.



(c) Mismatch severity at step 300.



(d) Mismatch severity at step 400.

Figure 7: Training–inference mismatch at steps 100–400 of Qwen3-1.7B (GRPO). Each row compares BF16, W4A16, W4A4 with CPA, and naive W4A4. CPA suppresses the growing mismatch under W4A4.

A.2 Controlled Mismatch Comparisons

We compare a fixed checkpoint on matched inputs without RL updates (Table 3), measuring cross-stack mismatch with `low_var_kl`. BF16 uses BF16 execution in both stacks; the quantized settings compare FSDP fake quantization (FQ) with vLLM real-FP4 execution. The last row instead uses identical FQ kernels across stacks to remove real-versus-fake kernel differences.

Table 3: Cross-stack mismatch at a fixed checkpoint. W/A denote weight/activation precision; relative values use the BF16 control (0.0011) as the baseline.

Setting	Cross-stack comparison	W	A	Mismatch KL	Relative
BF16	BF16 vs. BF16	BF16	BF16	0.0011	$1.0\times$
W4A16	FQ vs. real	FP4	BF16	0.0017	$1.5\times$
W4A4	FQ vs. real	FP4	FP4	0.047	$42.7\times$
W4A4	FQ vs. FQ (same kernel)	FP4	FP4	0.034	$30.9\times$

W4A16 stays close to BF16, while adding FP4 activation quantization raises mismatch to $42.7\times$ the BF16 value. Identical FQ kernels still produce $30.9\times$ mismatch across stacks, but zero within one stack. These controls support our diagnosis: activation FP4 amplifies cross-stack numerical differences, even without real-versus-fake kernel differences.

A.3 Large-Mismatch Tokens

In the inspected naive W4A4 Qwen3-1.7B GRPO trajectory, $|\Delta_t^{\mathrm{stack}}|$ (Equation (7)) is small before collapse (roughly the first 300 steps): the maximum is at most 1 and the mean at most 0.2 in log-probability units, with little positional concentration. After collapse, values can exceed 20 and concentrate at the sequence tail, particularly on function words such as `the` and `that`, rather than numerals or operators. These are qualitative observations; token identity alone does not determine mismatch. CPA keeps mismatch low and avoids collapse in the corresponding run.

B Experimental Settings

B.1 Training Recipes and Hyperparameters

Table 4 lists the settings for GRPO with Qwen3-1.7B-Base and DAPO with Qwen3-8B-Base. DAPO additionally uses accuracy-based group filtering and a 512-token overlong buffer with a penalty factor of 1.0.

Table 4: Experimental settings for the GRPO and DAPO recipes.

Parameter Name	**GRPO Recipe**	**DAPO Recipe**
`trainer.nnodes`	1	1
`trainer.n_gpu_per_node`	8	8
`data.train_batch_size`	256	32
`data.max_prompt_length`	1024	1024
`data.max_response_length`	512	8192
`rollout.n`	8	8
`rollout.temperature`	1.0	1.0
`rollout.top_p`	1.0	1.0
`val_kwargs.temperature`	0	0.6
`actor.ppo_mini_batch_size`	256	32
`optim.lr`	1×10^{-6}	1.5×10^{-6}
`optim.weight_decay`	0.01	0.1
`actor.use_kl_loss`	False	False
`rollout_correction.rollout_is`	token	token
`rollout_correction.rollout_is_threshold`	2.0	2.0

B.2 Hardware and Software Configuration

Table 5 summarizes the shared pipeline: real-NVFP4 rollout and fake-quantized training with BF16 master weights, colocated on one eight-GPU node.

Table 5: Hardware and software configuration of the experimental pipeline.

Component	Configuration
Compute node	One node with eight FP4 capable GPUs
GPU memory	180 GB per GPU
Peak Tensor Core throughput	BF16 / FP8 / FP4: 2.25 / 4.5 / 9 PFLOP/s
CUDA	13.0
Training framework	PyTorch 2.9.1 with FSDP2
Rollout engine	vLLM 0.15.0
vLLM memory utilization	0.80
W4A4 rollout GEMM path	FlashInfer–CUTLASS
Training master weights	BF16
Modules excluded from quantization	`lm_head` and `embed_tokens`
Resource placement	Training and rollout colocated on the same node

B.3 Choosing the Alignment Coefficient

We use $\lambda_{\text{CPA}} = 10^{-3}$ by default across the reported model scales and response lengths. It achieves the best average accuracy for `low_var_kl` under both GRPO and DAPO (Tables 6 and 7). As an empirical guideline, keep 10^{-3} while training-side mismatch KL remains low and stable; if it trends upward, consider increasing λ_{CPA} within the tested range $[10^{-3}, 10^{-2}]$. Monitor reward or accuracy alongside mismatch, since excessive alignment can slow RL improvement. This is a tuning heuristic, not an automatic adaptation rule.

C Detailed Ablation Results

Tables 6 and 7 detail the ablations in Section 4.4; `abs_logprob` corresponds to `mae`. "Avg." averages task scores within each row; "Avg. per type" further averages over the four coefficients.

Table 6: GRPO ablation results across different loss types and coefficient values.

Loss type	Coef.	GSM8K	MATH500	SVAMP	Avg.	Avg. per type
`low_var_kl`	0.001	86.1	60.2	88.0	78.1	77.6
	0.003	84.0	58.3	89.0	77.1	
	0.005	85.2	61.7	87.0	78.0	
	0.01	83.1	59.8	88.3	77.1	
`mse`	0.001	84.8	57.9	87.0	76.6	76.9
	0.003	84.1	59.4	89.0	77.5	
	0.005	83.2	61.3	87.3	77.3	
	0.01	81.8	59.9	86.6	76.1	
`abs_logprob`	0.001	79.4	57.0	40.0	56.8	70.3
	0.003	84.6	59.9	86.3	76.9	
	0.005	84.5	61.1	87.3	77.6	
	0.01	72.7	55.8	81.0	69.8	

Table 7: DAPO ablation results across different loss types and coefficient values.

Loss type	Coef.	AIME24	AIME25	AMC	MATH500	Avg.	Avg. per type
`low_var_kl`	0.001	30.3	20.9	59.8	77.6	47.1	45.8
	0.003	28.7	18.3	58.7	77.6	45.8	
	0.005	25.5	18.6	61.3	78.2	45.9	
	0.01	22.4	16.3	60.5	77.4	44.2	
`mse`	0.001	28.9	22.2	58.4	76.6	46.5	45.8
	0.003	25.6	18.4	60.1	77.0	45.3	
	0.005	29.2	20.1	59.6	77.8	46.7	
	0.01	25.8	18.4	56.4	78.0	44.7	
`abs_logprob`	0.001	28.5	19.9	59.7	77.6	46.4	45.6
	0.003	27.2	20.4	59.4	78.8	46.5	
	0.005	26.0	20.1	59.0	79.0	46.0	
	0.01	23.7	14.6	58.3	76.6	43.3	

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Abstract and section 1 accurately summarize the core contributions and scope of this work.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: This paper explicitly discusses the limitations of the proposed method in section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This work is empirical in nature and does not present formal theorems or proofs. All claims are supported by controlled experiments and ablations in Section 3.1 and Section 4.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All experimental settings, pipeline and key details are fully described in section 4 and section B to reproduce the main results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The training and evaluation code is not publicly available at this time. Experimental settings are described in section 4 and Appendix B.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Datasets, coefficient selection, and other hyperparameters are specified in section 4 and section B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We report mean accuracy over multiple sampled responses per prompt (section 4), but do not quantify variability across independent training runs due to computational budget constraints.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Experimental configurations and per-step runtime are reported in Appendix B and section 4.3, respectively.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research strictly complies with the official NeurIPS Code of Ethics throughout all procedures.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work is fundamental algorithmic research with no direct negative societal impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work does not release new datasets or model checkpoints requiring additional release safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All adopted datasets and open-source codes are properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new datasets, or model checkpoints are released with this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper involves no crowdsourcing experiments or human subject research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human participants are involved, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are only used for paper writing and polishing, not as core methodological components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.