WHERE ROUNDING HAPPENS MATTERS: ROUNDING-LOCATION ALIGNMENT STABILIZES LLM REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning for large language models often uses separate systems for inference and training. Even with identical weights and BF16 tensors, implementation differences can make an intended on-policy update off-policy and destabilize training. Existing remedies align the entire inference and training stacks, change global precision, or modify the learning algorithm. We ask whether aligning selected parts of the execution stack can stabilize BF16 RL while other implementation differences remain. We align training-side rounding locations with inference in three regions—RoPE, SwiGLU, and residual RMSNorm—by retaining selected intermediates in FP32 until their BF16 outputs. Across four Qwen-family models, aligning all three regions prevents the collapse observed under the default BF16 baseline; RoPE-path alignment alone also suffices in the studied runs, despite remaining numerical discrepancies. The interventions keep the inference engine, matrix-multiplication and attention kernels, and learning objective unchanged.

1 INTRODUCTION

Reinforcement learning (RL) has become a central method for post-training large language models (LLMs), improving mathematical reasoning, code generation, and search-assisted problem solving (Shao et al., 2024; Guo et al., 2025; Jin et al., 2025). Maintaining stable optimization remains difficult, however, and model performance can collapse during training (Yu et al., 2025; Zheng et al., 2025). Training–inference mismatch is one source of this instability (Yao et al., 2025; Zhong et al., 2026).

Many RL systems use an inference engine such as vLLM to generate responses and a separate implementation, such as Hugging Face Transformers, to recompute their probabilities for training (Kwon et al., 2023; Wolf et al., 2020). Even with the same checkpoint and nominal data type, implementation differences can make the recomputed probabilities disagree with those of the policy that generated the responses. This discrepancy turns an intended on-policy update off-policy and can destabilize training (Yao et al., 2025; Zhong et al., 2026; Qi et al., 2025).

Prior studies stabilize training under this mismatch through global precision changes (Qi et al., 2025) or algorithmic corrections (Yao et al., 2025; Zhong et al., 2026). Another approach seeks exact numerical agreement by aligning arithmetic and reduction order across engines (He & Thinking Machines Lab, 2025; Zhang et al., 2026b), including the precision of intermediate computations in RoPE and SwiGLU (Panda, 2026). We ask whether aligning only selected parts of the execution stack can stabilize BF16 RL while retaining the learning objective and allowing other implementation differences to remain.

A compiled/eager comparison motivates this question. Both compiled and eager execution support stable training when inference and training use aligned modes; the two crossed conditions collapse. This observation directs attention to agreement between the engines and motivates inspecting the numerical changes introduced by fusion. Separate kernels store some intermediates in BF16, whereas a fused kernel can retain them in FP32 until its output. We call this difference in where values are rounded a *rounding-location mismatch*.

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.

Guided by graph inspection, we align training-side rounding locations with those in inference in three regions: the RoPE path (Su et al., 2024), SwiGLU (Shazeer, 2020), and residual RMSNorm (Zhang & Sennrich, 2019). We retain selected training intermediates in FP32 and round each region's outputs to BF16, preserving BF16 model weights, the inference engine, matrix-multiplication and attention kernels, and learning objective.

Contribution. We show that selective rounding-location alignment can stabilize BF16 LLM RL without full-stack numerical agreement or additional mismatch corrections in the learning objective. Across four Qwen-family models, aligning all three regions prevents the collapse observed under the default BF16 baseline despite numerical disagreement; RoPE-path alignment alone also suffices in every studied run. These results establish selective alignment as a practical route to stable training in the studied settings.

2 Related Work

Corrections for training–inference mismatch. Truncated importance sampling and sequence-level trust-region masking limit the effect of distribution shift between inference and training policies (Yao et al., 2025; Li et al., 2026). Zheng et al. (2025) analyze how numerical disagreement and policy staleness affect token-level surrogate objectives. Other remedies reduce the learning rate when response length signals impending instability (Zhang et al., 2026a), or accept candidate updates using an inference-side discrepancy proxy (Liang et al., 2026).

Execution alignment. Batch-invariant kernels fix reduction structure to remove numerical dependence on batch size (He & Thinking Machines Lab, 2025); invariant reductions can also extend agreement across tensor-parallel sizes (Zhang et al., 2026b). Zhong et al. (2026) use the same Hugging Face implementation for inference and training with batch-invariant kernels. Their experiments show that numerical disagreement alone can cause training collapse. In mixture-of-experts models, replaying the expert selections made during inference reduces routing disagreement and stabilizes training (Ma et al., 2026). The Ring-linear report aligns training and inference module by module (Ling Team et al., 2025); Panda (2026) explicitly matches rounding locations in RoPE and SwiGLU as part of achieving bitwise-identical token log probabilities.

Precision and quantization. Qi et al. (2025) show that switching from BF16 to FP16 reduces training–inference mismatch and improves RL stability in their experiments. Low-bit RL systems also address the tradeoff between inference efficiency and numerical agreement. Huang et al. (2025) combine NVFP4 quantization with low-rank adaptation to accelerate rollouts and study quantization noise as a source of exploration. Other approaches combine FP8 linear-layer computation and KV-cache quantization with importance-sampling corrections (Qiu et al., 2026), or align the FP8 precision flow across training and rollouts to reduce numerical mismatch (Xi et al., 2026). QaRL aligns low-bit matrix multiplications in the training forward pass with quantized inference, together with importance correction and a modified policy objective (Gu et al., 2026). Related work on long-context training shows that BF16 errors can disrupt the relative-position property of rotary position embedding (RoPE) (Wang et al., 2025).

3 Training–Inference Mismatch in LLM RL

LLM RL commonly separates response generation from optimization. An inference engine such as vLLM generates responses token by token, using an implementation optimized for efficient generation. A training implementation such as Hugging Face Transformers then processes those same responses to recompute their token probabilities and differentiate the learning objective (Kwon et al., 2023; Wolf et al., 2020). At identical weights, these two implementations are intended to represent the same policy. Numerical differences can make their probabilities disagree before any optimization step, creating *training–inference mismatch* in addition to the policy changes caused by learning. This mismatch poses a major challenge for LLM RL, as it can destabilize training and cause performance collapse (Yao et al., 2025; Zhong et al., 2026; Qi et al., 2025).

3.1 WHERE THE MISMATCH COMES FROM

The same model can produce different floating-point results in different implementations. We distinguish differences in precision and casts, kernels, and reduction order.

Precision and casts. Parameter, buffer, computation, and accumulation precision can differ, as can cast locations. To accelerate generation, some inference engines quantize the KV cache or use low-precision computation, reducing memory traffic or arithmetic cost; these choices can introduce further disagreement with higher-precision training (Qiu et al., 2026; Xi et al., 2026). BF16 tensor storage also does not imply that every intermediate is computed in BF16. For two operations f and g, let R_{bf16} denote rounding to BF16. Rounding the intermediate and retaining it in FP32 give, respectively,

$$\underbrace{R_{\text{bf16}}(g(R_{\text{bf16}}(f(x))))}_{\text{intermediate rounded to BF16}} \quad \text{and} \quad \underbrace{R_{\text{bf16}}(g(f(x)))}_{\text{intermediate retained in FP32}} .$$

Both return BF16 outputs, but g can receive different values. We call this a *rounding-location mismatch*. Compilation can introduce this difference through *kernel fusion*, which combines operations in one kernel: a fused kernel can retain an intermediate in FP32, whereas separate kernels may store and reload it in BF16 (Ansel et al., 2024).

Kernels. The same operation can use different kernels or decompositions. vLLM combines Q, K, and V projections and gate and up projections, whereas Hugging Face may execute them separately; the systems can also use different attention implementations (Dao et al., 2022).

Reduction order. Floating-point addition is not associative, so batch size, tensor parallelism, and kernel tiling can change a reduction's result. Invariant kernels across batch and parallel sizes fix the reduction structure.

3.2 ALGORITHM AND EXPERIMENTAL SETUP

We define the sampling process, policy update, and discrepancy measure used in our experiments.

Policies and advantages. At the start of an iteration, let θ_0 denote the checkpoint used for inference. The inference engine uses policy μ_{θ_0} to sample G responses $y_i = (a_{i,1}, \ldots, a_{i,T_i})$ for a prompt $x \sim \mathcal{D}$. The training engine starts from $\theta = \theta_0$ and uses policy π_θ to recompute the token probabilities of those responses. Even at identical weights, the policies can differ numerically. Each response receives a correctness reward R_i, and its advantage is centered within the group without standard-deviation normalization:

$$A_i = R_i - \bar{R}, \qquad \bar{R} = \frac{1}{G}\sum_{j=1}^{G} R_j. \tag{1}$$

Objective. Prior work addresses inference–training mismatch with importance-sampling corrections, truncation, and masking rules (Yao et al., 2025; Li et al., 2026; Zheng et al., 2025). To study execution alignment under a simple learning objective, we use PPO-clipped GRPO (Schulman et al., 2017; Shao et al., 2024) without additional mismatch corrections, and hold this objective fixed across conditions. Let $s_{i,t} = (x, a_{i,<t})$ and define the training-side ratio $r_{i,t}(\theta) = \frac{\pi_\theta(a_{i,t}|s_{i,t})}{\pi_{\theta_0}(a_{i,t}|s_{i,t})}$. With expectation over the sampling process above, the update minimizes

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}_{\mu_{\theta_0}}\left[\frac{1}{G}\sum_{i=1}^{G}\sum_{t=1}^{T_i}\min\Big\{r_{i,t}(\theta)A_i, \text{clip}(r_{i,t}(\theta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})\,A_i\Big\}\right]. \tag{2}$$

The denominator, sampled responses, and advantages are fixed; gradients flow through π_θ. At the start of an update, $r_{i,t}(\theta_0) = 1$ because both probabilities use the training implementation, even if it disagrees with the inference policy that sampled the responses. Clipping therefore controls training-policy changes while leaving the initial discrepancy between μ_{θ_0} and π_{θ_0} uncorrected.

Mismatch measurement. To measure numerical disagreement, we compare the two implementations at the same fixed parameters θ_0. For tokens sampled from μ_{θ_0}, we use the following estimate of per-token forward Kullback–Leibler (KL) divergence (Kullback & Leibler, 1951):

$$\widehat{D}_{\mathrm{KL}} = \frac{1}{N}\sum_{i=1}^{G}\sum_{t=1}^{T_i}\left[\log \mu_{\theta_0}(a_{i,t} \mid s_{i,t}) - \log \pi_{\theta_0}(a_{i,t} \mid s_{i,t})\right], \qquad N = \sum_{i=1}^{G} T_i. \tag{3}$$

Experimental setup. The main-text RL experiments use Sanity-Test-R1D-1.5B, a MATH subset selected to be neither trivial nor unsolvable for the initial model (Hendrycks et al., 2021; Qi et al., 2025), and evaluate on problems from the 2025 American Invitational Mathematics Examination (AIME 2025) (Mathematical Association of America, n.d.). vLLM generates responses; Hugging Face Transformers recomputes their probabilities and updates the model, with fully sharded data parallelism (FSDP) (Zhao et al., 2023) as the distributed sharding strategy. Appendix A reports execution settings, sample counts, and optimization hyperparameters.

4 Motivation: Small Execution Changes Can Destabilize RL

We motivate the study with a small change to execution: enabling or disabling compilation during inference and training. We use DeepSeek-R1-Distill-Qwen-1.5B in BF16, following the experimental setup in Section 3.

We compare compiled and eager execution on the vLLM inference side and the Hugging Face training side. The four conditions are compiled–compiled, compiled–eager, eager–compiled, and eager–eager, where the first mode denotes inference. In the two aligned conditions, inference and training use the same execution mode; in the two crossed conditions, they use different modes. Appendix A gives the exact execution settings.

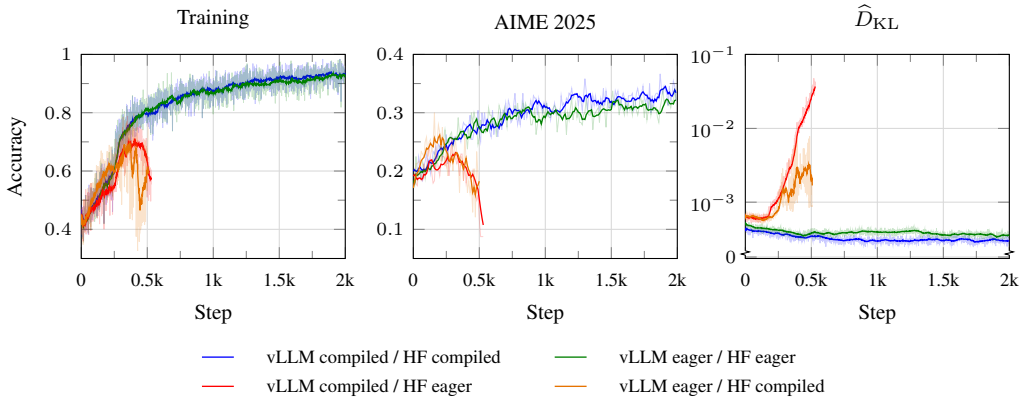

Figure 1: Training accuracy, AIME 2025 accuracy, and the inference–training discrepancy $\widehat{D}_{\mathrm{KL}}$ (Eq. 3) for the four execution-mode conditions. Observations are smoothed using causal trailing means over 20 observations for training accuracy and $\widehat{D}_{\mathrm{KL}}$ and five observations for AIME 2025 accuracy. Both aligned conditions maintain high accuracy; both crossed conditions initially improve and then lose accuracy.

Figure 1 shows that both compiled and eager execution support stable training when inference and training use aligned modes. The two crossed conditions collapse, accompanied by rising $\widehat{D}_{\mathrm{KL}}$, whereas both aligned conditions retain a small, nonzero discrepancy.

With eager training held fixed, changing inference from compiled to eager restores stability without increasing training-side precision. Together, the four conditions support agreement between execution modes as a stabilizing factor. They motivate inspecting the numerical changes introduced by compilation to identify rounding locations for targeted alignment.

5 QWEN ANALYSIS: WHERE FUSION MOVES ROUNDING LOCATIONS

We identify rounding-location mismatches by inspecting intermediate data types and fused operations in the inference and training graphs. The aim is to find values stored in BF16 in one implementation but retained in FP32 until a later operation in the other.

For Qwen2 and Qwen3 (Yang et al., 2024; 2025), inspection of the compiled and eager vLLM paths identifies three regions in which fusion changes rounding locations: Q/K normalization, when present, plus RoPE; SwiGLU's sigmoid linear unit (SiLU) activation (Elfwing et al., 2018) and gate multiplication; and residual addition plus root mean square normalization (RMSNorm).

Figure 2 shows the identified locations. Fused execution retains the marked intermediates in FP32 and stores region outputs in BF16. Separate operations round those intermediates to BF16 before passing them to the next operation.

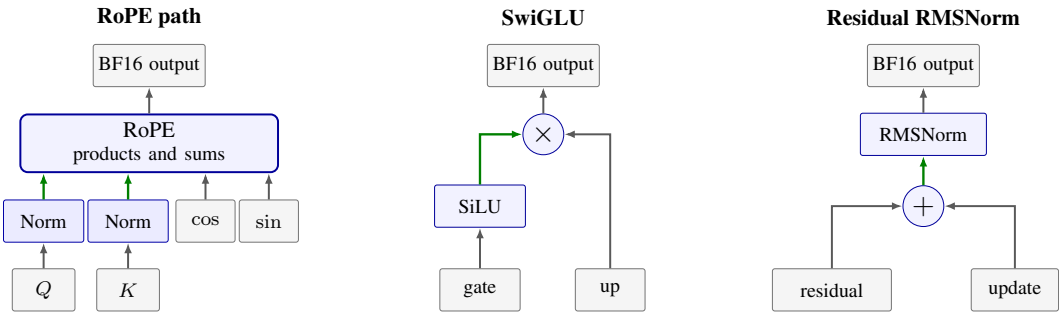

Figure 2: The three Qwen regions in which fusion changes rounding locations. Green arrows carry intermediates retained in FP32 under fusion and rounded to BF16 between separate operations. RoPE products and sums are also retained in FP32 under fusion. Region outputs are BF16 in both cases. Q/K normalization, when present, is part of the RoPE path.

5.1 THE THREE AFFECTED REGIONS

Q/K normalization and RoPE. In variants with Q/K normalization, fusion retains its statistics, normalized and scaled Q/K values, and subsequent RoPE products and sums in FP32. Variants without Q/K normalization still fuse RoPE products and sums. Only rotated Q/K outputs are stored in BF16.

SwiGLU. Fusion retains sigmoid and SiLU results in FP32 through multiplication with the up projection, then stores the product in BF16. Eager execution stores the SiLU output in BF16 before multiplication.

Residual RMSNorm. Under fusion, RMSNorm consumes the unrounded FP32 residual sum and retains its intermediate computations in FP32. Both its output and a separate copy of the sum for the continuing residual stream are stored in BF16. Eager execution rounds the sum before RMSNorm. This region follows both attention and MLP in every decoder block.

The intervention retains the identified training intermediates in FP32 and casts each region output back to BF16. Changes are confined to pointwise and normalization operations in the RoPE, SwiGLU, and residual-RMSNorm regions, preserving the inference engine and the compute-intensive matrix-multiplication and attention kernels. Aligning all three regions adds 5.33–10.92% to forward-and-backward latency, including communication, and 0.41–0.89 GiB of peak allocated memory per GPU across three benchmarked models (Section C). The additional training time is expected to represent a smaller percentage of total RL time once rollout generation is included.

6 Static Attribution of Policy Discrepancy

We quantify how aligning the three regions identified in Section 5 changes policy discrepancy at fixed model weights. In Hugging Face recomputation, the interventions keep in FP32, respectively: Q/K through RoPE (including normalization when present); SiLU output and gate multiplication; and residual addition and RMSNorm.

Each region output is cast back to BF16 at its original location; model weights, matrix multiplications, and attention kernels are unchanged. We compute all 2^3 combinations on fixed inference responses and log probabilities; only Hugging Face recomputation changes. For simplicity, the static analyses here and in Section 8 use three models, omitting Qwen3-4B-Base because its architecture closely resembles Qwen3-8B-Base. Table 1 reports the estimator in equation 3. Appendix A gives the prompt, response, and token counts.

Table 1: Estimated per-token KL divergence from vLLM to Hugging Face. Check marks indicate internal FP32 computation during recomputation; bold values are column minima. The first eight rows report $\widehat{D}_{\mathrm{KL}} \times 10^4$, and the final two give percentage reductions from the BF16 baseline.

Region computed in FP32			$\widehat{D}_{\mathrm{KL}} \times 10^4$		
RoPE path	SwiGLU	Residual RMSNorm	DeepSeek-R1-Distill-Qwen-1.5B	Qwen3-1.7B	Qwen3-8B-Base
			6.5416	7.4461	3.7599
✓			5.0951	7.1912	3.4247
	✓		6.1650	7.6227	4.1681
		✓	6.1040	7.4906	3.4921
✓	✓		5.0548	6.9537	3.4139
✓		✓	5.0716	6.8715	3.6027
	✓	✓	6.4336	7.7358	3.7176
✓	✓	✓	**4.8741**	**6.7689**	**3.2891**
RoPE-path reduction (%)			22.1	3.4	8.9
Three-region reduction (%)			25.5	9.1	12.5

Computing all three regions in FP32 yields the smallest estimate for every model. Relative to the all-BF16 condition, $\widehat{D}_{\mathrm{KL}}$ decreases by 25.5%, 9.1%, and 12.5% for the three models, respectively.

We use Shapley attribution (Shapley, 1953) to allocate the joint KL reduction among the three interventions, averaging each intervention's marginal contribution over all orders in which they can be applied (Appendix A.3). The RoPE path accounts for 81.5%, 91.8%, and 76.9% of the joint reduction. The SwiGLU shares are 8.6%, -3.8%, and -14.3%; the residual-RMSNorm shares are 9.9%, 12.0%, and 37.4%.

These shares describe average marginal contributions across combinations, not the fraction of the joint reduction recovered by each intervention alone. A negative share means that an intervention increases discrepancy on average; it does not by itself establish interaction effects. The nonmonotonic results show that higher intermediate precision does not automatically reduce mismatch. These fixed-weight comparisons identify the RoPE path as the largest contributor to the joint discrepancy reduction. Whether that reduction prevents training collapse is a separate question, tested next with RL runs.

7 Rounding-Location Alignment Stabilizes Training

The static attribution motivates testing both RoPE-path alignment and alignment of all three regions. We now evaluate whether these interventions stabilize RL training. We use the objective and data from Section 3 with four Qwen-family models spanning Qwen2 and Qwen3: DeepSeek-R1-Distill-Qwen-1.5B, Qwen3-1.7B, Qwen3-4B-Base, and Qwen3-8B-Base. Appendix A gives the sampling, optimization, and execution settings. We compare four conditions for each model:

- **BF16 baseline:** default vLLM inference and Hugging Face training.

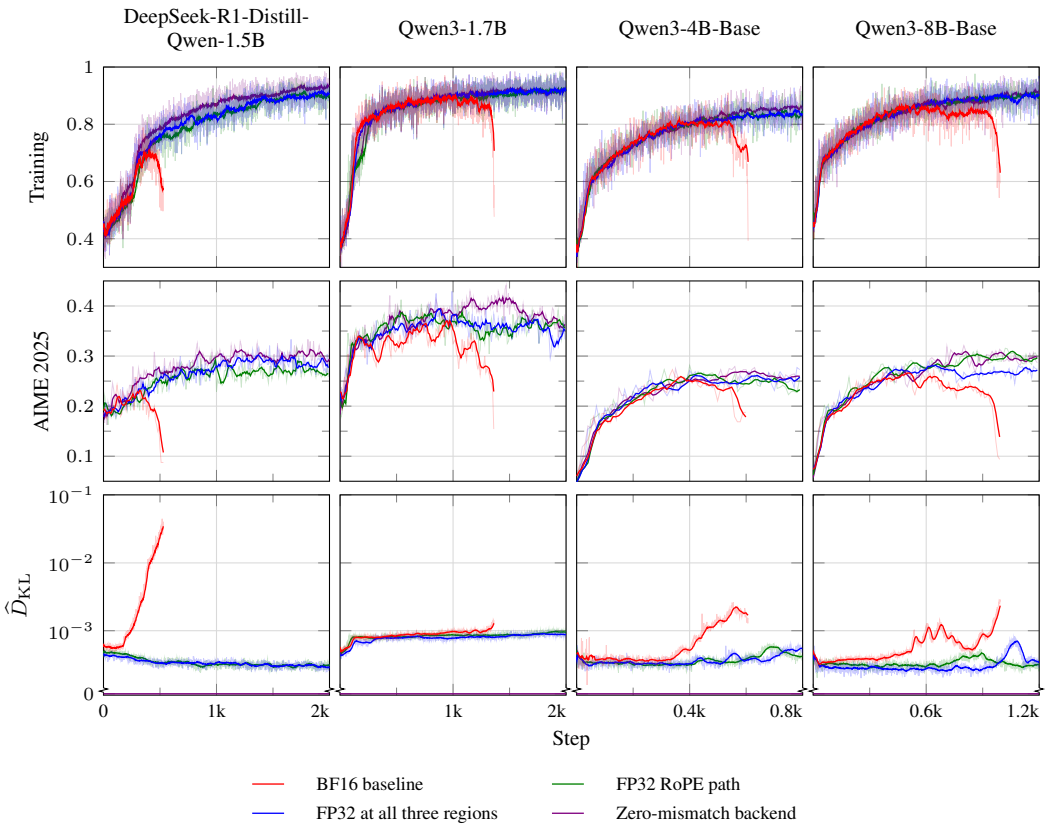

Figure 3: Training accuracy (top), AIME 2025 accuracy (middle), and inference–training discrepancy $\widehat{D}_{\mathrm{KL}}$ (bottom; Eq. 3). Observations are smoothed using causal trailing means over 20 observations for training accuracy and $\widehat{D}_{\mathrm{KL}}$ and five observations for AIME 2025 accuracy. The BF16 baseline collapses for every model; the FP32 RoPE path, FP32 at all three regions, and Zero-mismatch backend remain stable.

- **FP32 RoPE path:** only the training-side RoPE path retains FP32 intermediates, including Q/K normalization when present.
- **FP32 at all three regions:** the training-side RoPE, SwiGLU, and residual-RMSNorm regions retain FP32 intermediates.
- **Zero-mismatch backend:** the same Hugging Face model implementation is used for inference and training, with batch-invariant kernels. This aligns precision and casts, kernels, and reduction order. Section D describes the implementation.

Figure 3 shows that the BF16 baseline improves and then loses accuracy for every model. Both targeted alignments and the Zero-mismatch backend avoid collapse and retain higher AIME 2025 accuracy. $\widehat{D}_{\mathrm{KL}}$ rises as the baseline collapses, sharply for DeepSeek and more modestly for Qwen3-1.7B. Both targeted alignments retain small, nonzero $\widehat{D}_{\mathrm{KL}}$, while the Zero-mismatch backend remains at zero. These results show that inspecting the graphs and aligning the identified rounding locations greatly stabilizes BF16 RL. Retaining FP32 intermediates along the RoPE path alone is sufficient in all observed runs.

The Zero-mismatch backend confirms that the same objective can train stably with aligned execution. The two targeted conditions show that eliminating all numerical disagreement is unnecessary for stability in these runs: aligning the RoPE path alone suffices, with no additional correction in the learning objective. This is evidence about training stability, complementing the fixed-weight discrepancy reductions in Section 6. Appendix B repeats these controls with DeepSeek-R1-Distill-

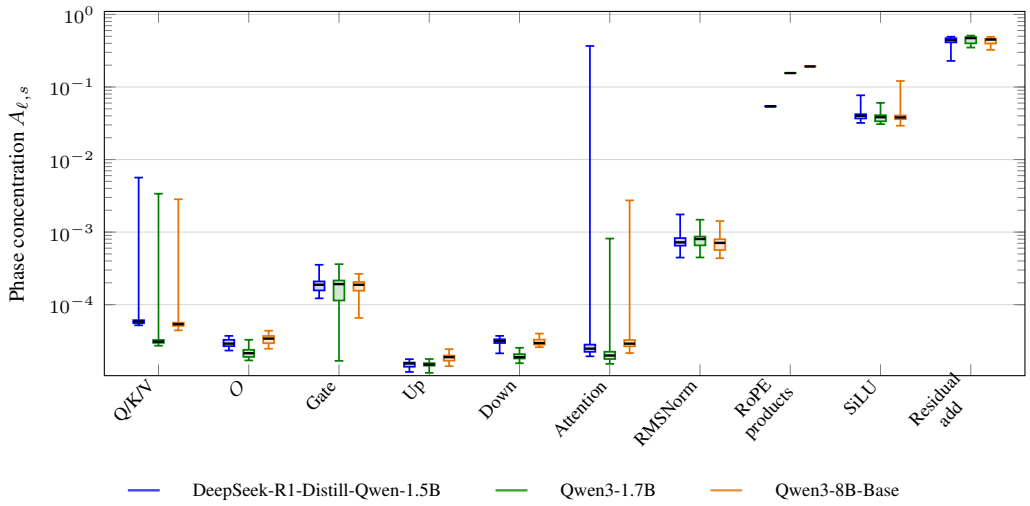

Figure 4: Layer distributions of the 16-frequency phase concentration. The center line is the layer median, boxes span the first and third quartiles, and whiskers show the full range. The three intermediates at changed rounding locations are markedly more nonuniform than the other signals.

Qwen-1.5B on DAPO-Math-17k (Yu et al., 2025) and likewise finds that RoPE-path alignment stabilizes training.

RoPE-path alignment also has lower overhead than aligning all three regions: it adds only 0.90–2.04% to training-computation latency, with unchanged peak allocated memory, in the three-model benchmark (Section C).

8 Structure of BF16 Rounding Phases

We explore the numerical structure of intermediates at the identified rounding locations. This characterization complements the policy-discrepancy and RL experiments by examining the values directly affected by rounding. A value's *BF16 cell phase* is its normalized position between adjacent BF16 values; away from midpoint ties, this position determines the direction and relative size of rounding error. Using the same three models as in Section 6, we compare RoPE products, SiLU outputs, and residual sums with projection, attention, and RMSNorm outputs.

For each signal s and layer ℓ, the phase-concentration statistic $A_{\ell,s}$ is the root mean square of the first 16 Fourier magnitudes of the phase distribution; larger values indicate stronger departures from uniformity over these frequencies. We report its median across layers, $\widehat{A}_s$. Section A.4 gives the sampling procedure, formal definitions, and individual frequency spectra.

Figure 4 compares the groups. Across models, $\widehat{A}_s$ is 1.5×10^{-5}–8.0×10^{-4} for projection, attention, and RMSNorm outputs, compared with 0.038–0.040 for SiLU, 0.054–0.192 for RoPE products, and 0.443–0.472 for residual sums. The three groups whose rounding locations change under fusion have the largest median phase concentrations. Some Q/K/V and attention layers have large maxima, but low medians and quartiles show that they are isolated.

The three regions identified by graph inspection thus exhibit distinctive BF16 cell-phase structure across all three models. The individual spectra also show consistent frequency patterns (Section A.4). These measurements quantify how values are distributed within BF16 intervals; their rounding errors and downstream effects additionally depend on interval widths and subsequent computations (Section A.5). The Fourier analysis provides a complementary view of the affected intermediates alongside the static discrepancy reductions and RL stabilization reported in Sections 6 and 7.

9 DISCUSSION

Why a small gap matters. Aligning all three regions reduces the static per-token KL by 9.1–25.5%, yet prevents collapse in the RL runs. One possible explanation is that numerical errors at different locations affect training updates differently. Aligning a sensitive region may therefore improve stability substantially while leaving much of the average policy discrepancy intact.

Why not simply compile training? Compilation is a possible remedy: the compiled–compiled condition is stable in the motivating experiment. It does, however, introduce practical costs. Hugging Face training compilation covers both forward and backward passes and incurs a substantial initial compilation cost (Hugging Face, n.d.). Variable response lengths can trigger recompilation, while data-dependent control flow and unsupported operations can cause graph breaks (PyTorch, 2025). Distributed sharding and custom kernels may also require compiler-specific integration. A reported large-scale training deployment required explicit work on graph breaks, custom kernels, and numerical agreement with eager execution (Team PyTorch at IBM & Team PyTorch at Meta, 2024). Compiling Hugging Face also does not make it the same implementation as compiled vLLM: precision and casts, kernels, and reduction order can still differ.

Alignment or increased training-side precision? With the eager training implementation and its precision held fixed, switching inference from compiled to eager execution restores stability (Section 4). Together with the crossed execution-mode controls, this supports execution alignment as a stabilizing factor and shows that stability can be restored without increasing training-side precision. Section 6 also shows that higher intermediate precision does not automatically reduce static policy discrepancy. Together with the targeted rounding-location interventions, these results support rounding-location alignment as a stabilizing mechanism.

Limitations. The experiments cover two math datasets and models from a single architectural family. Extending the intervention to other architectures requires inspecting their inference and training graphs to identify the relevant rounding locations and adapt the alignment accordingly. The results demonstrate the importance of rounding-location alignment for stable LLM RL; broader experiments are needed to establish when it is sufficient across architectures and training settings.

Takeaway. The results support inspecting the inference and training graphs to identify and align rounding locations moved by fusion as a targeted intervention for stabilizing BF16 RL. RoPE-path alignment alone suffices to stabilize the studied BF16 RL runs. Aligning all three regions stabilizes training and gives the lowest per-token KL among the conditions compared in Section 6. The interventions keep the inference engine, matrix-multiplication and attention kernels, and training objective unchanged. Stable training therefore does not require full-stack alignment in these experiments.

AI USE STATEMENT

Generative AI tools were used to help search for related work, polish the writing of this paper, and assist with coding. The authors take responsibility for the final content, including all claims, AI-assisted text, and code.

ETHICS STATEMENT

This work studies numerical stability in language-model reinforcement learning using existing models and mathematical reasoning benchmarks. Improved training stability may support more reliable experimentation, but does not establish that the resulting models are safe, fair, or suitable for deployment. The rounding-location alignment intervention does not address harmful outputs, biases inherited from pretrained models, or misuse of improved model capabilities. These concerns require separate assessment in downstream applications.

Reproducibility Statement

We provide supplementary code to reproduce the RL experiments in Sections 4 and 7. Section 3 specifies the learning objective and discrepancy measure. Sections 4 and 7 describe the execution-mode comparisons and rounding-location alignment controls, and Section 5 identifies the affected computation regions. Sections 6 and 8 define the static analyses. Appendix A collects the datasets, hardware, sampling and execution settings, RL hyperparameters, observation-selection procedure, and supporting mathematical details. Section D describes the shared model path and batch-invariant kernels used by the Zero-mismatch backend. These specifications document the conditions of the reported runs; the limitations discussed in the main text delimit the scope of the empirical conclusions.

References

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster machine learning through dynamic Python bytecode transformation and graph compilation. In *ASPLOS*, volume 2, pp. 929–947, 2024. doi: 10.1145/3620665.3640366.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *NeurIPS*, volume 35, pp. 16344–16359, 2022. doi: 10.52202/068431-1189.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. doi: 10.1016/j.neunet.2017.12.012.

Hao Gu, Hao Wang, Jiacheng Liu, Lujun Li, Qiyuan Zhu, Bei Liu, Binxing Xu, Lei Wang, Xintong Yang, Sida Lin, Sirui Han, and Yike Guo. QaRL: Rollout-aligned quantization-aware RL for fast and stable training under training–inference mismatch. *arXiv preprint arXiv:2604.07853*, 2026. URL https://arxiv.org/abs/2604.07853.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You,

Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Horace He and Thinking Machines Lab. Defeating nondeterminism in LLM inference. *Thinking Machines Lab: Connectionism*, September 2025. doi: 10.64434/tml.20250910. URL https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/. Accessed September 26, 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks Track*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Wei Huang, Yi Ge, Shuai Yang, Yicheng Xiao, Huizi Mao, Yujun Lin, Hanrong Ye, Sifei Liu, Ka Chun Cheung, Hongxu Yin, Yao Lu, Xiaojuan Qi, Song Han, and Yukang Chen. QeRL: Beyond efficiency – quantization-enhanced reinforcement learning for LLMs. *arXiv preprint arXiv:2510.11696*, 2025. URL https://arxiv.org/abs/2510.11696v1.

Hugging Face. torch.compile for training, n.d. URL https://huggingface.co/docs/transformers/en/torch_compile. Accessed September 26, 2026.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Ö. Arık, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *COLM*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. A study of BFLOAT16 for deep learning training. *arXiv preprint arXiv:1905.12322*, 2019. URL https://arxiv.org/abs/1905.12322.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. URL https://arxiv.org/abs/1412.6980.

S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. doi: 10.1214/aoms/1177729694.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *SOSP*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165.

Yingru Li, Jiacai Liu, Jiawei Xu, Yuxuan Tong, Ziniu Li, and Baoxiang Wang. Trust region masking for long-horizon LLM reinforcement learning. In *ICML*, 2026. URL https://openreview.net/forum?id=dPaBOu0Xwk.

Jing Liang, Hongyao Tang, Yi Ma, Yancheng He, Weixun Wang, Xiaoyang Li, Ju Huang, Wenbo Su, Jinyi Liu, Yan Zheng, Jianye Hao, and Bo Zheng. The mirage of optimizing training policies: Monotonic inference policies as the real objective for LLM reinforcement learning. *arXiv preprint arXiv:2606.29526*, 2026. URL https://arxiv.org/abs/2606.29526.

Ling Team, Bin Han, Caizhi Tang, Chen Liang, Donghao Zhang, Fan Yuan, Feng Zhu, Jie Gao, Jingyu Hu, Longfei Li, Meng Li, Mingyang Zhang, Peijie Jiang, Peng Jiao, Qian Zhao, Qingyuan Yang, Wenbo Shen, Xinxing Yang, Yalin Zhang, Yankun Ren, Yao Zhao, Yibo Cao, Yixuan Sun, Yue Zhang, Yuchen Fang, Zibin Lin, Zixuan Cheng, and Jun Zhou. Every attention matters: An efficient hybrid architecture for long-context reasoning. *arXiv preprint arXiv:2510.19338*, 2025. URL https://arxiv.org/abs/2510.19338.

Wenhan Ma, Hailin Zhang, Liang Zhao, Yifan Song, Yudong Wang, Fuli Luo, and Zhifang Sui. Stabilizing MoE reinforcement learning by aligning training and inference routers. In *ICML*, 2026. URL https://openreview.net/forum?id=ZMkfGplf4b.

Mathematical Association of America. MAA invitational competitions, n.d. URL https://maa.org/maa-invitational-competitions/. Accessed September 26, 2026.

Ashwinee Panda. 0 train-infer mismatch for open-weight MoE RL in open-source code, August 2026. URL https://kiddyboots216.github.io/mismatch/. Technical blog post, August 17, 2026. Accessed September 26, 2026.

PyTorch. torch.compile troubleshooting, 2025. URL https://docs.pytorch.org/docs/main/user_guide/torch_compiler/torch.compiler_troubleshooting.html. Last updated December 3, 2025. Accessed September 26, 2026.

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. URL https://arxiv.org/abs/2601.18150.

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.

L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker (eds.), *Contributions to the Theory of Games II*, volume 28 of *Annals of Mathematics Studies*, pp. 307–317. Princeton University Press, 1953. doi: 10.1515/9781400881970-018.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL https://arxiv.org/abs/2002.05202.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Team PyTorch at IBM and Team PyTorch at Meta. Maximizing training throughput using PyTorch FSDP and torch.compile, May 2024. URL https://pytorch.org/blog/maximizing-training-throughput/. Last updated November 13, 2024. Accessed September 26, 2026.

Philippe Tillet, H. T. Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019. doi: 10.1145/3315508.3329973.

Haonan Wang, Qian Liu, Chao Du, Tongyao Zhu, Cunxiao Du, Kenji Kawaguchi, and Tianyu Pang. When precision meets position: BFloat16 breaks down RoPE in long-context training. *Transactions on Machine Learning Research*, February 2025. URL https://openreview.net/forum?id=gwXfZ3xkUq.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *EMNLP: System Demonstrations*, pp. 38–45, 2020. doi: 10.18653/v1/2020.emnlp-demos.6.

Haocheng Xi, Charlie Ruan, Peiyuan Liao, Yujun Lin, Han Cai, Yilong Zhao, Shuo Yang, Kurt Keutzer, Song Han, and Ligeng Zhu. Jet-RL: Enabling on-policy FP8 reinforcement learning with unified training and rollout precision flow. *arXiv preprint arXiv:2601.14243*, 2026. URL https://arxiv.org/abs/2601.14243.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024. URL https://arxiv.org/abs/2407.10671.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 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 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, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *NeurIPS*, volume 38, pp. 113222–113244, 2025. doi: 10.52202/085713-3775.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Yaxiang Zhang, Yingru Li, Jiacai Liu, Jiawei Xu, Ziniu Li, Qian Liu, and Haoyuan Li. Beyond precision: Training-inference mismatch is an optimization problem and simple LR scheduling fixes it. *arXiv preprint arXiv:2602.01826*, 2026a. URL https://arxiv.org/abs/2602.01826.

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. In *ICML*, 2026b. URL https://openreview.net/forum?id=5eZmlUyFpl.

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. *Proceedings of the VLDB Endowment*, 16(12): 3848–3860, 2023. doi: 10.14778/3611540.3611569.

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. *arXiv preprint arXiv:2512.01374*, 2025. URL https://arxiv.org/abs/2512.01374v3.

Tianle Zhong, Neiwen Ling, Yifan Pi, Zijun Wei, Tianshu Yu, Geoffrey Fox, Peng Wu, and Xiao Yu. Diagnosing training inference mismatch in LLM reinforcement learning. *arXiv preprint arXiv:2605.14220*, 2026. URL https://arxiv.org/abs/2605.14220.

A Experimental Details

The main-text experiments use the Sanity-Test-R1D-1.5B training set and evaluate on AIME 2025. Unless a condition states otherwise, vLLM and Hugging Face use their default configurations: inference uses vLLM with FlashAttention, and Hugging Face training uses scaled dot-product attention without compilation. All experiments use the default sampling configuration: temperature 1.0, with no top-p or top-k filtering. Responses are truncated when they reach the configured length limit.

A.1 Static analyses

Section 6 uses the first 512 prompts, and Section 8 uses the first 256. In both analyses, we generate one response per prompt. Section 6 holds the responses and inference log probabilities fixed across all eight Hugging Face recomputations. Table 2 reports the sample sizes; token counts include generated response tokens only.

Table 2: Sampling configuration for Sections 6 and 8. Each section uses one response per prompt, so the prompt and response counts are equal.

Configuration	DeepSeek-R1-Distill-Qwen-1.5B	Qwen3-1.7B	Qwen3-8B-Base
Section 6			
Prompts	512	512	512
Responses	512	512	512
Response tokens	2,860,411	2,615,319	548,171
Response-token limit	10,240	5,120	5,120
Model-length limit	16,384	16,384	16,384
Section 8			
Prompts	256	256	256
Responses	256	256	256
Response tokens	1,050,873	1,308,191	270,510
Response-token limit	5,120	5,120	5,120
Model-length limit	16,384	16,384	16,384

A.2 RL training and evaluation

Table 3 summarizes the RL configuration used in Sections 4 and 7. Training metrics are logged once per step, and AIME accuracy is evaluated every 10 steps. At each AIME evaluation, every GPU independently generates one response for each question in the full test set. We compute accuracy on each GPU and report the mean across all GPUs.

For Section 4, compiled vLLM uses its default path, with `enforce_eager` set to `False`; eager vLLM sets it to `True`. Compiled Hugging Face training applies `torch.compile()` to every decoder layer; eager training executes the model directly.

A.3 Static attribution

For Section 6, let C be the set of three interventions and $D(S)$ the value of $\widehat{D}_{\mathrm{KL}}$ with subset S enabled. The Shapley contribution of region j to the reduction is

$$\Delta_j = \sum_{S \subseteq C \setminus \{j\}} \frac{|S|!(|C| - |S| - 1)!}{|C|!} \left[D(S) - D(S \cup \{j\}) \right]. \tag{4}$$

The reported shares are $\Delta_j / [D(\varnothing) - D(C)]$. They allocate the reduction achieved by the three interventions and sum to one. A negative share means that an intervention increases the discrepancy

Table 3: RL training and evaluation configuration.

Configuration	DeepSeek-R1-Distill-Qwen-1.5B	Qwen3-1.7B	Qwen3-4B-Base	Qwen3-8B-Base
GPUs	8	8	16	32
Prompts per step	64	64	64	64
Responses per prompt	8	8	8	8
Response-token limit	10,240	10,240	10,240	10,240
Evaluation model-length limit	16,384	16,384	16,384	16,384
Mini-batches per step	4	1	1	1
Optimizer	Adam (Kingma & Ba, 2015) $(\beta_1, \beta_2) = (0.9, 0.95)$			
Learning rate	10^{-6}			
PPO clipping $(\epsilon_{\text{low}}, \epsilon_{\text{high}})$	$(0.2, 0.28)$			

on average across the orders in which the interventions are applied. A share is not the fraction of the joint reduction obtained by applying that intervention alone, and a negative share does not by itself establish interaction effects.

The token log-probability differences in Section 6 can also be summed within each response. For the baseline conditions, the mean of these response-level log-probability differences is 3.65, 3.80, and 0.40 nats for DeepSeek-R1-Distill-Qwen-1.5B, Qwen3-1.7B, and Qwen3-8B-Base, respectively. These values equal $\widehat{D}_{\text{KL}}$ multiplied by the mean response length. They illustrate how small per-token differences accumulate over a response.

A.4 BF16 CELL PHASES AND FOURIER DIAGNOSTICS

For Section 8, a *signal* collects scalars from an FP32 intermediate where eager execution would store it in BF16. We compare ten groups: Q/K/V projection outputs; the attention output projection (O); gate, up, and down projection outputs; attention output; RMSNorm output; four RoPE products; SiLU output; and two residual sums per decoder block. Fusion changes rounding locations in the final three groups. Each is computed in FP32 from operands observed during the BF16 forward pass, not a separate full-FP32 model. Each scalar at selected generated-token positions is a sample; prompt and padding positions are excluded, layers are separate, and no nonfinite elements were observed. Table 2 reports the prompt, response, and token counts for DeepSeek-R1-Distill-Qwen-1.5B, Qwen3-1.7B, and Qwen3-8B-Base.

For a nonzero, finite FP32 value x in the normal range of BF16, let $e(x) = \lfloor \log_2 |x| \rfloor$. BF16 has 7 explicit fraction bits (8-bit significand precision) (Kalamkar et al., 2019), so adjacent values within $[2^{e(x)}, 2^{e(x)+1}]$ are spaced by $h(x) = 2^{e(x)-7}$. Define the cell phase

$$\phi(x) = \text{frac}(x/h(x)), \qquad 0 \le \phi(x) < 1, \tag{5}$$

where $\text{frac}(z) = z - \lfloor z \rfloor$; set $\phi(0) = 0$.

For $n_{\ell,s}$ sampled elements x_m of signal s in layer ℓ, compute

$$\widehat{c}_{\ell,s,k} = \frac{1}{n_{\ell,s}} \sum_{m=1}^{n_{\ell,s}} \exp(2\pi i k \phi(x_m)), \qquad k = 1, \ldots, 16. \tag{6}$$

Summarize the first 16 frequencies by

$$A_{\ell,s} = \left(\frac{1}{16} \sum_{k=1}^{16} |\widehat{c}_{\ell,s,k}|^2 \right)^{1/2}. \tag{7}$$

The statistic $A_{\ell,s}$ measures phase concentration over these frequencies. We summarize layers by

$$\widehat{A}_s = \text{median}_\ell \, A_{\ell,s}. \tag{8}$$

Figure 4 shows the distribution of $A_{\ell,s}$ across layers. Figure 5 reports the layer medians of $|\widehat{c}_{\ell,s,k}|$; for residual addition, the two residual sums' magnitudes are averaged within each layer before taking the median.

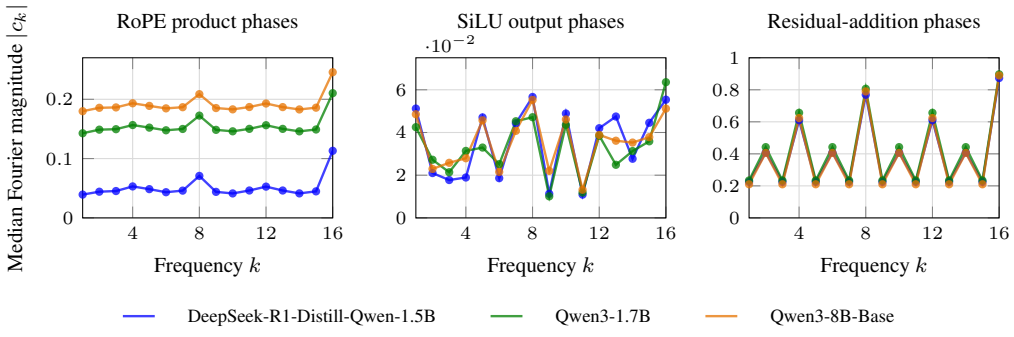

Figure 5: Fourier magnitudes for RoPE products, SiLU outputs, and residual sums. Each point summarizes $|\widehat{c}_{\ell,s,k}|$ by its median across layers. For residual addition, the magnitudes of the attention and MLP residual sums are averaged within each layer before taking the median. The panels use different vertical scales.

Figure 5 shows consistent frequency patterns across models. RoPE products have peaks at $k = 4, 8, 16$, and SiLU shows a similar alternating pattern. Residual sums have larger magnitudes at even frequencies, with further increases at multiples of four and eight and the largest magnitude at $k = 16$. These spectra characterize phase distributions; they do not establish a relationship between a particular frequency and RL stability.

A.5 Rounding error and BF16 cell phase

For the spacing $h(x)$ and phase $\phi(x)$ defined in Section A.4, round-to-nearest-even conversion R_{bf16} gives, away from midpoint ties and overflow,

$$R_{\mathrm{bf16}}(x) - x = h(x)\rho(\phi(x)), \qquad \rho(u) = \begin{cases} -u, & 0 \le u < \frac{1}{2}, \\ 1-u, & \frac{1}{2} < u < 1. \end{cases} \tag{9}$$

At a midpoint, the parity of the adjacent BF16 significands determines the rounding direction. Uniform phases independent of $h(x)$ give zero mean rounding error because $\int_0^1 \rho(u)\,du = 0$. Uniformity is sufficient under this independence assumption, but it is not necessary: if every value is exactly BF16-representable, then $\phi(x) = 0$ and the rounding error is zero, even though all Fourier coefficients equal one. Phase concentration therefore does not determine the mean rounding error. Effects on subsequent computations also depend on their sensitivity to the rounded values.

B Additional Experiment on DAPO-Math-17k

We repeat the four conditions in Section 7 with DeepSeek-R1-Distill-Qwen-1.5B, replacing the training dataset with DAPO-Math-17k (Yu et al., 2025). The PPO-clipped GRPO objective, sampling configuration, optimization hyperparameters, and execution settings are the same as for this model in Appendix A and Table 3; we also evaluate on AIME 2025.

Figure 6 shows training and AIME 2025 accuracy and the inference–training discrepancy $\widehat{D}_{\mathrm{KL}}$ (Eq. 3). Both targeted alignments and the Zero-mismatch backend sustain training progress, whereas the BF16 baseline loses accuracy after initial improvement. The baseline's $\widehat{D}_{\mathrm{KL}}$ rises sharply; both targeted alignments retain small, nonzero $\widehat{D}_{\mathrm{KL}}$, and the Zero-mismatch backend remains at zero. RoPE-path rounding-location alignment thus also stabilizes training on this additional dataset.

C Training Overhead of Rounding-Location Alignment

We compare the BF16 baseline with FP32 intermediates along the RoPE path and at all three regions (Section 5). Experiments use eight GPUs with fully sharded data parallelism (FSDP) in Hugging Face. Each GPU processes one random 5,120-token sequence per step. All conditions use FP32

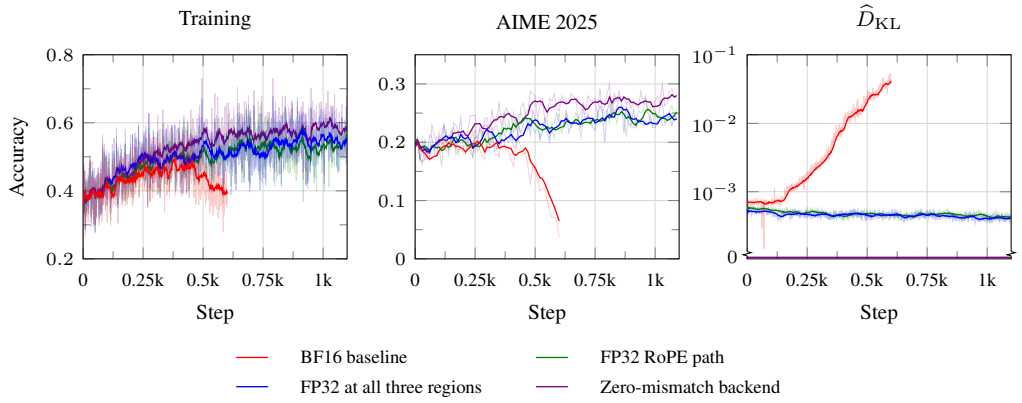

Figure 6: Training accuracy, AIME 2025 accuracy, and $\widehat{D}_{\mathrm{KL}}$ (Eq. 3) for DeepSeek-R1-Distill-Qwen-1.5B trained on DAPO-Math-17k. Observations are smoothed using causal trailing means over 20 observations for training accuracy and $\widehat{D}_{\mathrm{KL}}$ and five observations for AIME 2025 accuracy. The BF16 baseline loses accuracy after initial improvement; both rounding-location alignments and the Zero-mismatch backend sustain training progress.

master weights and gradient reduction, BF16 compute parameters, scaled dot-product attention, and non-reentrant gradient checkpointing; compilation and TF32 are disabled.

Timing covers forward, full-vocabulary FP32 log-softmax and mean next-token negative log likelihood over all prediction positions, and backward, including communication. Pretrained weights remain fixed; rollout, optimizer updates, data loading, and gradient clearing are excluded. Each run uses ten warmup and twenty measured steps. Step latency is the maximum wall-clock duration across ranks, measured after a barrier with device synchronization before and after each timed step. Table 4 reports latency changes averaged over paired run ratios and peak allocated memory maximized over ranks and runs.

RoPE-path alignment adds 0.90–2.04% latency with unchanged peak allocated memory; three-region alignment adds 5.33–10.92% latency and 0.41–0.89 GiB per GPU.

Table 4: Training-computation overhead. Latency is the mean step latency; changes are relative to each model's BF16 baseline.

Model	FP32 regions	Latency (ms/step)	Latency change (%)	Memory (GiB/GPU)
DeepSeek-R1-Distill-Qwen-1.5B	None (baseline)	702.61	—	12.62
	RoPE path	713.76	+1.59	12.62
	All three	779.31	+10.92	13.03
Qwen3-1.7B	None (baseline)	878.04	—	12.36
	RoPE path	895.96	+2.04	12.36
	All three	946.70	+7.82	12.91
Qwen3-8B-Base	None (baseline)	3025.63	—	19.00
	RoPE path	3052.81	+0.90	19.00
	All three	3186.77	+5.33	19.89

D Implementation of the Zero-Mismatch Backend

The Zero-mismatch backend in Section 7 uses the same uncompiled Hugging Face forward implementation for inference and training, preserving BF16 rounding locations. We implement batch-invariant kernels in Triton (Tillet et al., 2019) for matrix multiplication, log-softmax, mean reduc-

tion, and attention. Our implementation is based on the code released by He & Thinking Machines Lab (2025).

The matrix and reduction kernels keep accumulation order independent of batch size; log-softmax computes log probabilities in FP32, and mean reduction covers normalization statistics. Inference and training also share the same attention kernel and key/value traversal order, with FP32 accumulation and BF16 outputs. Cached decoding retains this reduction scheme without separately reducing partitions of the key/value sequence, aligning attention arithmetic with full-sequence recomputation.