DROPOUT-GRPO: VARIATIONAL ATTENTION DROPOUT FOR CONTINUOUS LATENT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models(LLMs) have achieved considerable success in various reasoning tasks by utilizing Chain-of-Thought(CoT) reasoning with intermediate language tokens. An emerging family of models utilize the latent continuous space of the LLMs in pursuit of robustness and efficiency, bypassing the restrictions of linguistic fluency and verbosity. However, often such models are deterministic by design and require complex architectural modifications to achieve variation in the reasoning trajectory necessary for critic-free post-training methodologies such as GRPO. We investigate dropout as a lightweight mechanism for exploring alternative latent trajectories without redesigning the recurrence. By applying dropout to attention probabilities, we perturb the continuous latent trajectories. We use rollout-specific attention-mask patterns shared across latent steps and replay the corresponding dropout realization during policy updates. This supplies rollout diversity without changing the latent recurrence architecture. We evaluate this approach on arithmetic benchmarks and support it with a mathematical analysis of replay and the scope of the resulting policy update. The results validate that latent reasoning models can benefit from post-training through reinforcement learning with verifiable rewards by using dropout. The corresponding code is shared via supplementary materials, and the repository will be made public upon acceptance of the paper.

1 INTRODUCTION

The emergence of Chain-of-Thought (CoT) reasoning (Wei et al., 2022; Zhou et al., 2022; Feng et al., 2023; Kojima et al., 2022) has significantly advanced the reasoning capabilities of Large Language Models (LLMs) (Comanici et al., 2025). In token-based CoT, each intermediate reasoning step is expressed through discrete language tokens by decoding a token from its hidden representation and adding that token as input to the next step, grounding reasoning to the generation of explicit text. Continuous latent reasoning models (Hao et al., 2024; Zhang et al., 2025) instead perform intermediate computation in a continuous representation space. Figure 1 illustrates the direct hidden-state feedback used by COCONUT-style models. The output state is fed back as the next input without selecting and re-embedding an intermediate vocabulary token, and only for the final answer is text decoding resumed. This distinction motivates latent recurrence as a way to perform reasoning without verbalizing every intermediate step, which allows the full dedication of compute budget to the reasoning process and exclude tokens for linguistic fluency.

The central challenge in latent reasoning is the transition of the model's semantic knowledge from a discrete language space to a continuous latent space. Various strategies have been explored to bridge this gap, including curriculum learning, knowledge distillationDeng et al. (2023), and structural layer modification (e.g., re-usingDu et al. (2025); Saunshi et al. (2025) or skippingChen et al. (2025); Cheng & Van Durme (2024) specific LLM layers). A prominent example of such models is COCONUT Hao et al. (2024), which leverages curriculum learning to internalize reasoning steps into implicit CoTDeng et al. (2024). While COCONUT demonstrates impressive logical reasoning, empirical observations suggest its mathematical capabilities can regress during curriculum training, highlighting a critical opportunity for post-training reinforcement learning (RL) to refine its performance.

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.

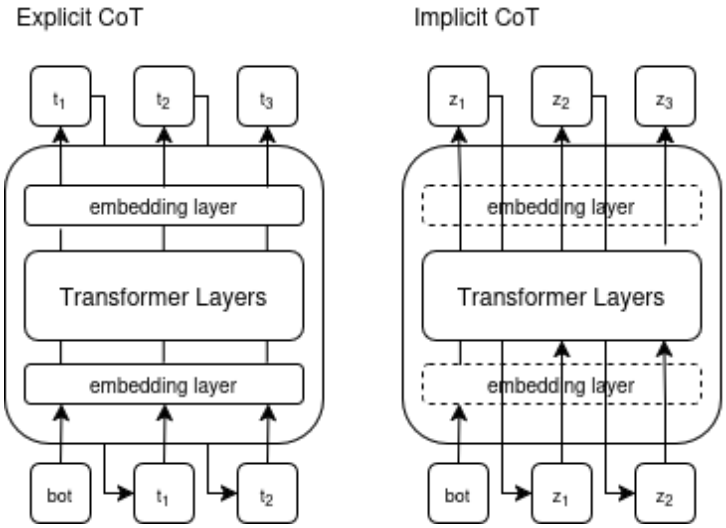

Figure 1: Token-based and continuous latent reasoning. Left: discrete tokens t_i are decoded and re-embedded for subsequent autoregressive steps. Right: continuous states z_i are fed directly into the Transformer during latent reasoning, bypassing intermediate token decoding and re-embedding (dashed boxes). Both paths start from an embedded boundary token (`bot`); the latent model returns to token decoding when generating its answer.

Reinforcement learning from verifiable rewards has established itself as the dominant post-training recipe for reasoning language models on mathematics and code benchmarks Shao et al. (2024); Yu et al. (2025). Among these methods, Group Relative Policy Optimization (GRPO) Shao et al. (2024) has emerged as a particularly lightweight and effective choice. It eliminates the learned value network of standard PPO Schulman et al. (2017) by estimating the per-prompt advantage from a small group of rollouts using a simple group-mean baseline.

However, applying GRPO to latent reasoning models presents a fundamental structural challenge. Token-level decoders naturally derive the necessary rollout diversity from stochastic sampling over the next-token distribution. In contrast, the continuous latent recurrence in models like COCONUT is inherently deterministic given the prompt and parameters. Consequently, multiple rollouts of the same prompt yield identical latent trajectories. With deterministic answer decoding, this also produces identical rewards and zero group-relative advantages. Stochastic answer decoding can reintroduce some answer diversity, but it cannot influence the preceding latent trajectories.

Our contribution. To bridge this gap, we introduce stochasticity through attention dropout, by using Bernoulli masks to perturb post-softmax attention probabilities. We keep attention dropout active during prompt processing, latent reasoning, and answer generation. Each rollout uses its own mask patterns, shared across latent steps on overlapping attention positions. During the policy update, we replay the corresponding dropout masks to reproduce the computation that generated each answer. Our analysis explains why dropout replay preserves rollout likelihoods and clarifies which practical training choices fall outside the policy-gradient guarantee. While modern LLM training has largely abandoned dropout as a regularizer, we observe that its ability to produce replayable structured perturbations makes it a natural mechanism for a source of stochasticity. We summarize our contributions as follows:

1. **A new paradigm for RL on latent-reasoning models.** We distinguish deterministic latent trajectories from the zero-advantage failure under deterministic answer decoding, and propose attention-dropout exploration with trajectory-consistent replay. This is the first method, to our knowledge, that successfully applies group-relative reinforcement learning to a continuous latent-reasoning model that uses hidden states directly in COCONUT style.
2. **A mathematical account of the practical method.** We explain why replay must preserve the dropout realization that generated each answer, and identify the assumptions under which an

idealized reward-gradient identity holds. The appendices give the full analysis, including the effects of response-length normalization.

3. **Empirical validation on GSM8K.** Dropout-GRPO improves a COCONUT baseline from 27.29% to 29.39% pass@1, demonstrating that the proposed construction yields a viable learning signal in the regime where deterministic-rollout GRPO fails entirely.

2 RELATED WORK

Explicit reasoning and Chain-of-Thought. Chain-of-Thought (CoT) prompting (Wei et al., 2022; Kojima et al., 2022) substantially improves LLM performance on symbolic, logical, and mathematical reasoning by articulating intermediate steps in natural language. The autoregressive generation of these steps incurs severe latency and memory-bandwidth overhead, motivating techniques such as CoT distillation (Hsieh et al., 2023) and prompt compression (Jiang et al., 2023). These approaches still operate within the discrete-token bottleneck, motivating paradigms that bypass language generation during reasoning entirely.

Latent-space reasoning. Hao et al. (2024) introduced COCONUT, providing a pioneering large-scale demonstration that continuous hidden states could replace discrete chain-of-thought tokens for multi-step reasoning. Subsequent work has explored the theoretical training dynamics of continuous CoT, framing it as a superposition of reasoning tracesDeng et al. (2025); Zhu et al. (2025). Other approaches have investigated hybrid discrete-continuous schemes via self-distillationShen et al. (2025) and explored policy optimization for latent tokensYue et al. (2025); Gozeten et al. (2025). Most relevant to our setting is LEPOZhou et al. (2026), which sources stochasticity by applying a Gumbel-softmax over vocabulary-embedding similarity scores at each latent step. While the relaxation of tokens over vocabulary embedding simplex has shown to be an effective source of stochasticity required for reinforcement learning, it re-imposes a discrete vocabulary bottleneck onto the latent state at an $\mathcal{O}(|V|)$ computational cost per step. Our approach instead introduces stochasticity through attention-probability masks that may vary across latent steps, with the corresponding execution noise replayed during policy updates. This construction acts through the masked network computation and does not require sampling a vocabulary-based latent token or identifying dropout with a weight perturbation, at the additional benefit of not requiring architectural modification or additional computational costs.

Reinforcement learning for language models. Policy-gradient training of language models begins with REINFORCE-style estimators (Williams, 1992) and was scaled to modern LLMs by Proximal Policy Optimization (PPO) (Schulman et al., 2017) with a learned value baseline. GRPO (Shao et al., 2024) removes the value model in favor of a group-mean baseline computed across K rollouts of the same prompt, and has demonstrated its efficiency and effectiveness in math reasoning post-training. Recent refinements include DAPO (Yu et al., 2025), which introduces "dynamic sampling" to filter zero-variance groups, and "Dr. GRPO" (Liu et al., 2025b), which observes that the standard within-group standard-deviation normalization $1/(\sigma_r + \delta)$ biases the policy gradient and recommends its removal. Inspired by DAPO's treatment of zero-variance groups, we mask these groups during loss computation and use an empirical group-accuracy filter to avoid unnecessary replay and backpropagation. Our GRPO updates use the Dr. GRPO mean-centered advantage without within-group standard-deviation normalization.

Variational dropout and Bayesian deep learning. Gal & Ghahramani (2016) establish that dropout, applied with a single mask held constant across an entire forward pass, can be interpreted as variational inference over a structured posterior on weights. We build directly on this interpretation: by enforcing a shared mask across all T latent recurrence steps within one rollout, each rollout can be interpreted as a single posterior sample, and the marginal policy as a Bayesian model average. The stochastic-computation-graph framework of Schulman et al. (2015) provides the broader setting for combining score-function terms for sampled outputs with differentiation through conditional computations.

3 DROPOUT-GRPO: EXPLORATION AND REPLAY

Figure 2 contrasts standard COCONUT with Dropout-GRPO. Dropout-GRPO enables native attention dropout to explore different latent trajectories, then uses verifier rewards to compare answers within each rollout group. During the policy update, it replays the corresponding dropout noise to score each answer under the computation that generated it. We first explain this source of exploration, then describe replay and the training loss. Formal assumptions and derivations are outlined in Appendices B–D.

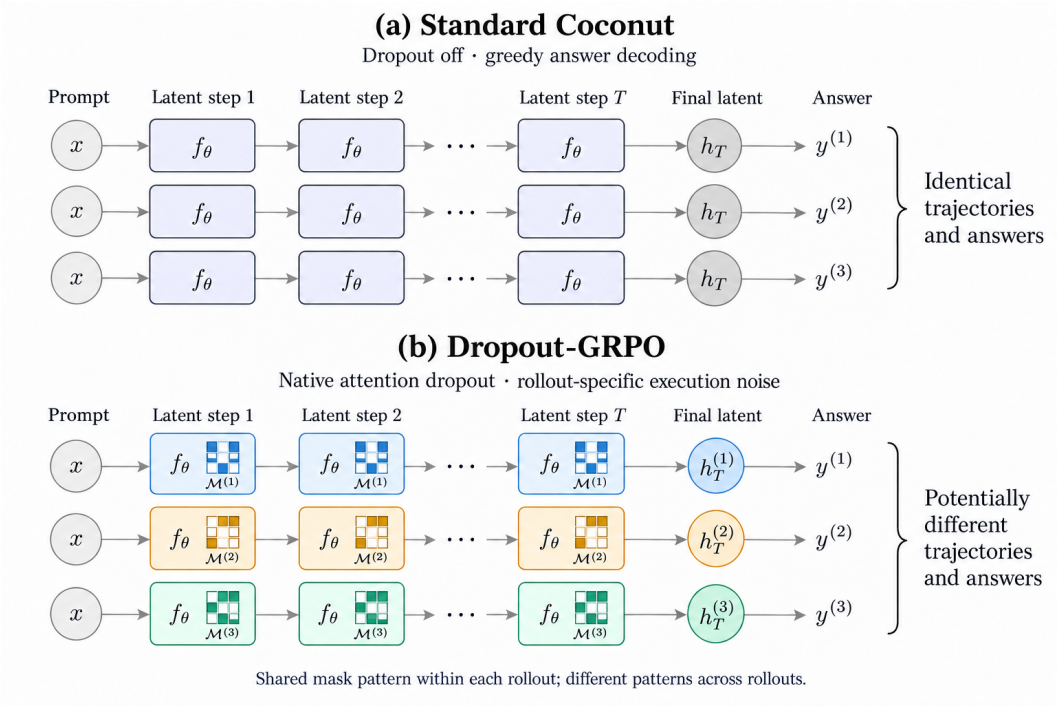

Figure 2: **Standard COCONUT versus Dropout-GRPO rollouts.** A standard Coconut architecture (top) is compared with the Dropout-GRPO rollout. Without noise supplied by dropout, the same prompt produces identical latent trajectories. The attention dropout perturbs the computation in each rollout, allowing different latent states and answers. $M^{(n)}$ denotes the shared mask pattern within rollout n, with different patterns across rollouts. All rollouts share parameters θ.

3.1 EXPLORING LATENT TRAJECTORIES WITH ATTENTION DROPOUT

For each prompt, a COCONUT-style model performs T continuous latent steps before decoding an answer. Each step processes the latest hidden state together with the causal prefix stored in the key/value cache. Without noise, the latent trajectory is fixed by the prompt and model parameters. Sampling answer tokens can diversify completions, but does not change the preceding latent trajectory within that rollout group.

We enable native attention dropout to perturb this computation. If A is the attention-probability matrix after softmax and V contains the value vectors, each attention call computes

$$O = \left(\frac{M \odot A}{1-p}\right) V, \qquad M_{ij} \sim \text{Bernoulli}(1-p). \tag{1}$$

where p is the dropout rate. The mask removes individual attention contributions before value aggregation, and does not select vocabulary tokens or mask hidden-state channels. The resulting perturbations can propagate through subsequent latent steps and diversify the reasoning trajectories without changing the model architecture.

Dropout remains active during prompt processing, latent recurrence, and answer decoding. Each rollout has its own dropout realization ξ, which specifies a collection of layer- and head-specific masks. Within a latent chain, mask decisions are shared across steps on overlapping attention positions. During the update, we replay the entire dropout realization, including the prompt and answer computations. Appendix C.1 describes the execution schedule and its backend dependence.

3.2 Replaying the computation for policy updates

We generate K answers per prompt and score them using the task verifier. The trainer records CPU and CUDA RNG states at rollout-chunk boundaries. During the update, it restores these states and recomputes the latent trajectory and answer likelihoods using the same batch structure, call schedule, and fixed answer tokens. We write this conditional policy as $\pi_\theta(y \mid x, \xi)$. The trainer accumulates gradients over chunks and takes one optimizer step per rollout batch. At each step, the dropout masks are checked for constancy.

3.3 Group-relative training objective

For answer k with reward $r^{(k)}$, we use the mean-only advantage $A^{(k)} = r^{(k)} - K^{-1}\sum_j r^{(j)}$, without dividing by the group's reward standard deviation (Liu et al., 2025b). Let $\rho_{k\ell}$ be the ratio of the replayed and old-policy probabilities for answer token ℓ, both conditioned on the rollout noise and answer prefix. Writing the clipped GRPO term as $c(\rho, A) = \min\{\rho A, \mathrm{clip}(\rho, 1-\varepsilon, 1+\varepsilon)A\}$, the minimized loss is

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{L_k}\sum_{\ell=1}^{L_k}\left[-c(\rho_{k\ell}, A^{(k)}) + \beta H(u_{k\ell})\right]. \tag{2}$$

Here N is the number of retained answers, L_k their scored lengths, β the regularization strength, and $u_{k\ell} = \log \pi_{\mathrm{ref}} - \log \pi_\theta$ at the scored token. Advantages and generated answers are held fixed during differentiation. The corrected regularizer uses $k_3(u) = e^u - u - 1$, or its Huber-style extension H with linear tails beyond $\pm\delta$; we use $\delta = 5$. The extension avoids clamping away tail gradients, but changes the penalty and is not an unbiased KL statistic. Appendix C specifies the full loss and Algorithm 1.

With answers sampled from the scored policy, exact replay, and the stated regularity conditions, an idealized unfiltered *sequence-sum* reward surrogate has the expected-reward gradient up to the group-baseline factor $1 - 1/K$ at the generating parameters. The proof and sequence-mean bias calculation are in Appendix B.

Answer sampling. During training, the latent model samples answer tokens from its conditional policy rather than selecting the highest-probability token. Attention dropout provides additional exploration in the latent computation. Replay scores the stored sampled answer tokens under the corresponding dropout realization; it does not sample a new answer during the update.

4 Experiments

4.1 Model

We use `Qwen2.5-1.5B` (Yang et al., 2024) as the base model for all experiments. We deviate from the GPT-2-based backbone of the original COCONUT paper (Hao et al., 2024) for two reasons: (i) GPT-2 uses absolute positional embeddings, which combined with left-padding induces a position bias that is particularly harmful in the latent-recurrence regime where prompt length varies across rollouts; (ii) Qwen2.5-1.5B provides an optimal balance by retaining modern architectural choices (RoPE, GQA, SwiGLU) and sufficient representational capacity for policy learning, while remaining tractable on a single GPU for the $K \cdot T$ rollout compute that GRPO requires.

We compare the following models:

- **Qwen2.5-1.5B base** (zero-shot and 1-shot).

- **Qwen2.5-1.5B + Coconut SFT.** COCONUT-style supervised fine-tuning following the curriculum of Hao et al. (2024), at latent depth $T = 6$. Curriculum training uses $c = 3$ latent tokens per reasoning step.
- **Coconut + GRPO (deterministic rollouts).** Standard GRPO without dropout. Expected to fail by the argument of Section 1; included to demonstrate the structural barrier.
- **Coconut + REINFORCE + dropout (EMA baseline).** Single rollout per prompt ($K = 1$) with a running EMA baseline in place of the group-mean baseline. Isolates the contribution of group-relative structure from the contribution of latent stochasticity.
- **Coconut + dropout-GRPO (ours).** The full method of Algorithm 1.

4.2 DATASETS

The dataset used and the respective number of prompts are detailed in Table 1. The updated proposed-method results in Table 3 use a GRPO training set that excludes GSM-Hard and SVAMP, removing the contamination sources identified in the earlier training mixture. The retained GRPO source is OpenMathInstruct-2; SVAMP is used only for evaluation in this revised setup. The choice of SFT dataset is consistent with the baseline COCONUT model, which provides concise per-step intermediate reasoning suitable for the COCONUT curriculum. We deliberately match the SFT and GRPO data distributions in style and difficulty. The COCONUT latent representation proved to be sensitive to the prompt distribution in our experiments, and large distributional shift between SFT and GRPO degraded the SFT-initialized latent representation faster than GRPO can productively refine it. Concise-step math-reasoning datasets in the GSM8K-AUG domain are scarce, which limits the scope of dataset-side ablations.

4.3 TRAINING DETAILS

Hyperparameters are detailed in Table 2. We train for 3000 GRPO steps with a batch size of 16 prompts and $K = 32$ rollouts per prompt, dropout rate $p = 0.1$, peak learning rate $\eta_{\max} = 10^{-5}$ with 200-step warmup and cosine decay, KL coefficient $\beta = 5 \times 10^{-3}$, group filter window $[\alpha_{\min}, \alpha_{\max}] = [0.20, 0.95]$, and Huber threshold $\delta = 5$.

We load and train the model weights in bfloat16 precision and optimizer states in 8-bit, to maximize group size K in a given compute budget. All reported experiments were conducted on a single GPU. Our implementation was based on the COCONUT code released by the original authors, and includes the DDP-coordinated skip mechanism described in Appendix C.4 as a forward-compatible hook for multi-rank training, but multi-GPU operation has not been empirically validated.

Table 1: Training data sources and volume

Dataset	Usage Vol.
GSM8K-aug (SFT)	385620
OpenMathInstruct-2 (GRPO)Toshniwal et al. (2024)	69981
GSM8K (Testing)	1319
MultiArith (Testing)Roy & Roth (2015)	600
SVAMP (Testing)	300

Table 2: Training hyperparameters.

Parameter	Value
Base model	Qwen Base
Latent depth T	6
Dropout rate p	0.1
Group size K	32
Train batch size (prompts)	16
Peak learning rate $\eta_{\max}$	1×10^{-5}
Warmup steps	200
Total GRPO steps	3000
KL coefficient β	5×10^{-3}
PPO clip ε (lower / upper)	0.10 / 0.12
Filter window $[\alpha_{\min}, \alpha_{\max}]$	$[0.20, 0.95]$
Huber threshold δ	5
Reference renewal interval	500 steps

Table 3: Accuracy with pass@1 across model variants. Boldface indicates the best result. The Coconut baseline uses a single seed due to high computational costs in training; other experiments use 3 seeds.

Model	GSM8K	SVAMP	MultiArith
Qwen2.5-1.5B (zero-shot)	62.47%	43.00%	82.50%
Coconut SFT (baseline)	27.29%	41.67%	61.67%
Coconut + GRPO(deterministic rollout)	27.29%	41.67%	61.67%
Coconut + REINFORCE + dropout (EMA)	$25.09\% \pm 0.39\%$	$38.78\% \pm 0.84\%$	$57.39\% \pm 0.70\%$
Coconut + dropout-GRPO (ours)	$\mathbf{29.39\% \pm 0.16\%}$	$\mathbf{44.00\% \pm 0.58\%}$	$\mathbf{64.67\% \pm 1.01\%}$

4.4 Results

We report the best post-warmup (step ≥ 200) validation pass@1 accuracy for all methods. Table 3 details these results across multiple test sets, evaluated without dropout. Dropout-GRPO achieves 29.39% pass@1, outperforming the Coconut-SFT baseline (27.29%) by 2.10 points. This marks the first successful application of group-relative reinforcement learning to a continuous latent-reasoning model.

On SVAMP, Dropout-GRPO improves accuracy from 41.67% to $44.00\% \pm 0.58\%$, a gain of 2.33 percentage points over Coconut SFT and 1.00 point over the Qwen2.5-1.5B base model. On MultiArith, accuracy increases from 61.67% to $64.67\% \pm 1.01\%$, a 3.00-point gain over Coconut SFT.

Figure 3 illustrates the training trajectories. Standard GRPO with deterministic rollouts yields a flat trajectory at the SFT initialization, confirming the structural barrier noted in Section 1. Without variance ($\sigma_r \equiv 0$), the per-rollout advantage $A^{(k)}$ and policy gradient vanish. Only the weak KL anchor pull remains, resulting in no measurable learning signal for Coconut-style latent reasoning.

Conversely, REINFORCE+dropout fails due to high variance. While latent stochasticity ensures diverse rollouts, replacing the group-relative baseline with a running EMA ($\alpha = 0.95$) degrades the gradient signal-to-noise ratio. At $K = 1$, the single-rollout gradient variance exceeds the policy's stability margin, destabilizing the SFT initialization faster than it can learn (visible in the early decline in Figure 3). This highlights that GRPO's group baseline is essential for variance reduction.

Finally, Figure 4 validates the necessity of within-group reward standard deviation ($\sigma_r > 0$) for non-trivial GRPO gradient flow. Without dropout, all K rollouts are bit-identical, collapsing σ_r to exactly zero and triggering the structural failure mode our work addresses. The REINFORCE ablation ($K = 1$) leaves within-group variance undefined, leading to the aforementioned destabilization. Ultimately, our attention dropout successfully supplies the rollout diversity required by GRPO, whereas removing it entirely eliminates the gradient signal.

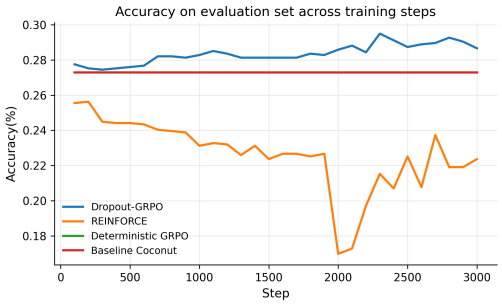

Figure 3: Evaluation accuracy across training steps



Figure 4: σ_r within prompt groups over training. Rollout diversity only occurs during Dropout-GRPO.

Component ablations and evaluation with dropout enabled or disabled are reported in Appendix F.

5 Discussion and Limitations

The experiments establish feasibility at the Qwen2.5-1.5B scale on the reported arithmetic benchmarks. While successful, the 2.10-percentage-point GSM8K gain is modest, and its behavior at larger scales remains to be established. Evaluation is constrained by the concise reasoning data available for COCONUT SFT, and extending the method to harder tasks requires broader and careful data collection and evaluation.

The experimental update and deployment policy differ from the idealized analysis, as explained in Section 3.3. In particular, sequence-mean aggregation, outcome-based filtering, and sampled regularization prevent a general unbiased-gradient claim. Dropout-free evaluation measures the deployed model directly, not the dropout-averaged training objective.

Architecturally, modern decoder architectures (Qwen2/LLaMA/Mistral/Gemma) typically expose dropout only in attention layers, and dropout is generally inactive during their own pre-training or SFT.Liu et al. (2025a) Our construction therefore reactivates a mechanism the field has largely retired for regularization. We use dropout here as a source of structured exogenous stochasticity rather than as a regularizer. The proof framework of Section B still survives the attention-only restriction because it requires only that ξ be exogenous and replayable.

Attention dropout operates during prompt processing and answer decoding as well as latent recurrence. The experiments therefore do not isolate which phase supplies the useful perturbations, nor establish that latent-phase noise outperforms answer sampling alone. Within-chain mask consistency was verified on overlapping attention positions in the tested BF16 SDPA release configuration, and seed resetting does not guarantee this behavior for other attention backends or precisions. The Bayesian interpretation is a modeling assumption that was made to simplify the analysis. In practice, Monte-Carlo dropout proves to work just as well as a variational dropout, although our results were marginally better for the latter.

6 Conclusion

Dropout-GRPO uses native attention dropout to explore continuous latent trajectories using rollout-specific mask patterns shared within latent chains, and replays the corresponding dropout masks when scoring answers for policy updates. This provides a simple and practical method for verifier-based post-training of COCONUT-style models without redesigning the latent recurrence architecture. On GSM8K, the reported runs improve the SFT baseline from 27.29% to 29.39% pass@1. The mathematical analysis explains the role of replay and the scope of an idealized reward-gradient identity.

References

Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. *arXiv preprint arXiv:2502.13842*, 2025. URL https://arxiv.org/abs/2502.13842.

Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024. URL https://arxiv.org/abs/2412.13171.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025. URL https://arxiv.org/abs/2507.06261.

Jingcheng Deng, Liang Pang, Zihao Wei, Shicheng Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. Llm latent reasoning as chain of superposition. *arXiv preprint arXiv:2510.15522*, 2025. doi: 10.48550/arXiv.2510.15522. URL https://arxiv.org/abs/2510.15522.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023. URL https://arxiv.org/abs/2311.01460.

Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024. URL https://arxiv.org/abs/2405.14838.

Hanwen Du, Yuxin Dong, and Xia Ning. Latent thinking optimization: Your latent reasoning language model secretly encodes reward signals in its latent thoughts. *arXiv preprint arXiv:2509.26314*, 2025. URL https://arxiv.org/abs/2509.26314.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 70757–70798, 2023.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059. PMLR, 2016. URL https://proceedings.mlr.press/v48/gal16.html.

Halil Alperen Gozeten, M. Emrullah Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. *arXiv preprint arXiv:2505.23648*, 2025. doi: 10.48550/arXiv.2505.23648. URL https://arxiv.org/abs/2505.23648.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024. URL https://arxiv.org/abs/2412.06769.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics (ACL)*, pp. 8003–8017, 2023. doi: 10.18653/v1/2023.findings-acl.507. URL https://aclanthology.org/2023.findings-acl.507.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.825. URL https://aclanthology.org/2023.emnlp-main.825/.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 22199–22213, 2022. URL https://arxiv.org/abs/2205.11916.

Houjun Liu, John Bauer, and Christopher D. Manning. Drop dropout on single epoch language model pretraining. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2157–2166, Vienna, Austria, 2025a. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.111. URL https://aclanthology.org/2025.findings-acl.111/.

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

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752. Association for Computational Linguistics, 2015. doi: 10.18653/v1/D15-1202. URL https://aclanthology.org/D15-1202/.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*, 2025. URL https://arxiv.org/abs/2502.17416.

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

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, 2015.

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

Zhihong Shao et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025. doi: 10.48550/arXiv.2502.21074. URL https://arxiv.org/abs/2502.21074.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*, 2024. doi: 10.48550/arXiv.2410.01560. URL https://arxiv.org/abs/2410.01560.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 24824–24837, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992. doi: 10.1007/BF00992696.

An Yang et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. doi: 10.48550/arXiv.2412.15115. URL https://arxiv.org/abs/2412.15115.

Qiying Yu et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. URL https://arxiv.org/abs/2503.14476.

Zhenrui Yue, Bowen Jin, Huimin Zeng, Honglei Zhuang, Zhen Qin, Jinsung Yoon, Lanyu Shang, Jiawei Han, and Dong Wang. Hybrid latent reasoning via reinforcement learning. *arXiv preprint arXiv:2505.18454*, 2025. URL https://arxiv.org/abs/2505.18454.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *arXiv*, 2025. doi: 10.48550/arxiv.2505.15778.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022. URL https://arxiv.org/abs/2205.10625.

Yuyan Zhou, Jiarui Yu, Hande Dong, Zhezheng Hao, Hong Wang, Jianqing Zhang, and Qiang Lin. Lepo: Latent reasoning policy optimization for large language models. *arXiv preprint arXiv:2604.17892*, 2026. URL https://arxiv.org/abs/2604.17892.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *arXiv*, 2025. doi: 10.48550/arxiv.2505.12514.

AI USE STATEMENT

In this work, we used generative AI tools for revising the initial draft written by the authors, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assist in the writing of proofs, refine hypothesisimplementing the RNG seed caching and recovery of variational masks, drafting the mathematical proof section, interpreting results, and for proving the mathematical claim. We have not used generative AI tools for developing theoretical models or conceptual frameworks, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset, or support qualitative and thematic data analysis, and generating synthetic datasets is not applicable to this work. Additionally, we used generative AI tools for creating and editing model figures and images, summarize or analyze existing literature, brainstorming, editing the paper to improve readability, identifying relevant literature, and formatting references. We have reviewed all AI-assisted work. We checked the literature suggested by the LLM, curated and proofread the code generated meticulously, implemented tests and guardrails to prevent errors, and proofread the mathematical claims to the best of our abilities. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

A Extended Discussion and Future Work

A.1 Interpreting the performance gain from Dropout-GRPO

The empirical improvement of 2.10 percentage points on GSM8K, with a reported variation of 0.16 percentage points, is modest in absolute terms. The result should be read as a *viability* demonstration rather than as a competitive accuracy claim: prior to this work, group-relative reinforcement learning on Coconut-style latent reasoning produced no gradient signal whatsoever, so accuracy improvements on the order of the SFT baseline's noise floor are the relevant comparison. We expect the gap to widen as the backbone is scaled, latent depth T is extended, and SFT initializations are made cleaner.

A.2 Latent reasoning vs. token chain-of-thought

Pure-latent reasoning models continue to trail strong token-level chain-of-thought systems on math benchmarks, and our work does not claim otherwise. The contribution is structural: a key tool of the post-training toolkit (group-relative reinforcement learning) is now accessible to latent-reasoning models, which removes a barrier to closing the gap. Whether the gap closes in practice is an empirical question for future work.

A.3 Evaluation scope and dataset availability

Coconut curriculum training requires intermediate reasoning chains in a concise per-step format that allows the latent recurrence to absorb each step into a single hidden-state update. GSM8K-Aug (Deng et al., 2024) provides this for grade-school arithmetic, but comparable resources for advanced mathematics (MATH, MATH-500, AIME) are not currently available: the prevailing CoT-augmented variants of these datasets emit multi-paragraph derivations, which the Coconut latent representation cannot ingest at the curriculum step granularity.

We therefore restrict evaluation to benchmarks distributionally aligned with GSM8K-Aug (GSM8K, SVAMP, MultiArith). Extending Dropout-GRPO to harder benchmarks is a complementary direction that depends on either (i) curating concise-step variants of advanced math datasets, or (ii) extending the Coconut SFT methodology to ingest verbose intermediate reasoning, the latter of which aligns with A.2. Neither is a modification to the RL method introduced in this paper.

A.4 On the use of dropout in modern LLM training

The attention-dropout construction of Appendix B.1 reactivates a mechanism that has fallen out of favor in modern large-language-model training: contemporary foundation models, including Qwen2 (Yang et al., 2024) and LLaMA, are typically pre-trained and supervised-fine-tuned without dropout engaged.Liu et al. (2025a) The convention shift is well-motivated: dropout's classical role as a generalization regularizer is largely subsumed by data abundance at the scales these models are trained on, where the inductive bias from billions of training tokens supplies the regularization that Bernoulli masking once provided.

We emphasize that our use of dropout is functionally distinct from this regularization role. We do not engage dropout to improve generalization, but as a structured source of *exogenous stochasticity*. The operator's ability to produce replayable perturbations indexed by stored RNG information makes it a natural off-the-shelf instance of the execution-noise formulation in Appendix B. The shared-mask release schedule is one instance of this formulation; Theorem 3 also permits other temporal noise schedules and does not depend on a posterior interpretation.

Implications for transferability. The proof framework of Appendix B accommodates parameter-independent execution noise, subject to its smoothness and sampling assumptions and the matched-computation conditions for replay. This includes activation masks without a weight-space interpretation.

Dropout is the cleanest instance available in current decoder configurations, but the construction itself is mechanism-agnostic. Re-introducing residual or MLP dropout at fine-tuning time, or substituting alternative perturbation methodologies are natural directions for extending the present work to

architectures and training regimes where attention dropout alone provides insufficient stochasticity surface.

A.5 Future work

Several directions appear promising. First, extending dropout-GRPO to chain-of-thought datasets at larger scale, where the supply of high-quality SFT data is abundant. Second, scaling the backbone: $1.5\mathrm{B}$ parameters is small for math benchmarks, and the gain from RL post-training typically widens with model size. Third, multi-GPU validation and throughput: although the implementation includes a DDP-coordinated skip mechanism intended to handle asymmetric filter outcomes across ranks, this code path has not been empirically tested in a multi-GPU configuration. Validating correctness, measuring the all-reduce overhead per step, and tuning for throughput at scale are part of this future-work direction. Fourth, while pure-latent reasoning currently trails token-level chain-of-thought on math, the availability of group-relative reinforcement learning, which is enabled by the construction proposed in this paper, may be a key ingredient in closing that gap.

A.6 Computational filtering

The group-accuracy filter was introduced to reduce training computation, including redundant policy-gradient calculations for groups with identical rewards and hence zero advantages. The broader accuracy window also skips some groups with nonzero policy contributions. Moreover, skipping KL calculations or averaging only over retained groups can change the update even when the omitted policy advantages are zero. We therefore treat the filter as a practical computational heuristic, rather than an operation that preserves the original gradient. Filtering is excluded from the mathematical analysis: Theorem 3 concerns i.i.d. rollout groups from the stated policy, without outcome-based selection, and does not establish unbiasedness of the filtered implementation. We make no claim that filtering improves variance or signal-to-noise.

Formal assumptions, estimator identities, and implementation details are given in Appendices B and C.

B Mathematical Analysis

We first fix notation for the Coconut latent recurrence, GRPO, and the attention-dropout construction (Appendix B.1); we then establish well-definedness of the latent gradient (Appendix B.2), a scaled policy-gradient identity for an idealized sequence-sum surrogate and the bias introduced by the implemented sequence-mean aggregation (Appendix B.3), and conditional-likelihood consistency under mask replay (Appendix B.4). The analysis uses an augmented policy with exogenous execution noise. It includes the shared-mask release schedule as a special case, while allowing more general temporal dependence; a Bayesian posterior interpretation is not an assumption of the results.

B.1 Setup

Coconut latent recurrence. Given a prompt x and parameters $\theta \in \mathbb{R}^D$, a causal Transformer retains the entire prefix through its key/value cache. We therefore use the complete continuous state $S_t = (h_t, C_t) \in \mathbb{R}^{d_t}$, where h_t is the latest hidden vector and C_t contains all layerwise cached keys and values for the prompt and processed latent positions. Position indices and causal masks are fixed by the execution schedule and are implicit in the state. The dimension d_t may grow with the prefix length. The initial prompt pass is $S_0 = G_\theta(x)$; each latent step is a full Transformer evaluation $S_t = F_{t,\theta}(S_{t-1})$, including cache updates, rather than a map of h_{t-1} alone. Answer decoding conditions on S_T and the answer prefix. A verifier supplies $r(y, x) \in [0, 1]$ on answer exact-match. Without dropout, the latent state is deterministic conditional on x and θ, while sampled answer tokens can still differ across rollouts. Identical answers and zero group-relative advantages follow only when answer decoding is also deterministic.

GRPO. For a group of K rollouts under θ_{old} with rewards $r^{(k)} = r(y^{(k)}, x)$, the per-rollout advantage is $A^{(k)} = r^{(k)} - \mu_r$ with $\mu_r = \frac{1}{K}\sum_k r^{(k)}$; following Liu et al. (2025b) we omit the

$1/(\sigma_r + \delta)$ normalization for reasons given in Remark 5. The clipped per-token surrogate (Shao et al., 2024) is

$$\mathcal{L}^{\mathrm{GRPO}}(\theta) \;=\; \mathbb{E}_{x,\mathcal{G}} \frac{1}{K} \sum_{k=1}^{K} \frac{1}{|y^{(k)}|} \sum_{\ell} \min\Big(\rho_{\theta,\ell}^{(k)} A^{(k)},\; \mathrm{clip}(\rho_{\theta,\ell}^{(k)}, 1 \pm \varepsilon)\, A^{(k)} \Big) - \beta\, \mathcal{R}_{\theta_{\mathrm{old}}}(\theta), \quad (3)$$

with per-token ratio $\rho_{\theta,\ell}^{(k)} = \pi_\theta(y_\ell^{(k)} \mid \cdot)/\pi_{\theta_{\mathrm{old}}}(y_\ell^{(k)} \mid \cdot)$ and reference policy π_{ref}. Here $\mathcal{R}_{\theta_{\mathrm{old}}}(\theta)$ is the expected sequence-mean, sampled regularizer specified in Appendix C.2, with the old-policy sampling law held fixed; it is not identified with exact KL. Here $x \sim \mathcal{D}$ and $\mathcal{G} \sim \Pi_{\theta_{\mathrm{old}}}^{\otimes K}(\cdot \mid x)$ denotes a rollout group; the sampling law, observed lengths, and advantages are held fixed when differentiating the surrogate. Equation (3) uses the sequence-mean aggregation of the implementation. Theorem 3 instead analyzes an explicitly defined sequence-sum surrogate without the length divisor.

Attention dropout and the augmented policy. Let $\xi \sim p(\xi)$ be a parameter-independent random tape for a rollout, including all randomness used by attention dropout during prompt processing, latent recurrence, and answer decoding. The execution schedule maps this random tape to a collection of layer- and call-specific masks. For the shared-mask release schedule, masks agree on overlapping attention positions across latent steps. The formulation also permits other temporal dependence; it does not assume fresh independent masks at each step. A sufficiently long tape can be sampled before execution, even when generation stops at a random answer length. We condition on this tape and write the complete-state recurrence as

$$S_0 = G_\theta(x; \xi), \qquad S_t = F_{t,\theta}(S_{t-1}; \xi), \qquad t = 1, \ldots, T, \quad (4)$$

Thus each rollout follows a conditional computation determined by its execution noise. Including ξ in the augmented action gives the joint distribution over (ξ, y) conditional on x:

$$\Pi_\theta(\xi, y \mid x) \;=\; p(\xi)\, \pi_\theta(y \mid x, \xi), \qquad \pi_\theta(y \mid x, \xi) \;=\; \prod_{\ell} \pi_\theta\big(y_\ell \,\big|\, x, S_T(\theta, \xi, x), y_{<\ell}, \xi\big), \quad (5)$$

with marginal policy $\bar{\pi}_\theta(y \mid x) := \mathbb{E}_\xi[\pi_\theta(y \mid x, \xi)]$, the dropout-averaged policy. Our theoretical reward objective is $J(\theta) := \mathbb{E}_{x \sim \mathcal{D}}\, \mathbb{E}_{(\xi,y) \sim \Pi_\theta}[r(y, x)]$, which by construction equals the expected reward of $\bar{\pi}_\theta$; the practical sequence-mean update is a biased surrogate for its gradient.

Source and scope of stochasticity. For one attention head and call, with attention probabilities $A = \mathrm{softmax}(QK^\top/\sqrt{d_h} + B)$ and causal bias B, Equation (1) gives the masked attention output, with Bernoulli retention probability $1 - p$. The masks act on attention probabilities, before value aggregation and output projection; surviving rows are not renormalized to sum to one. This perturbs continuous states without projecting them into a vocabulary mixture. Subsequent computations can propagate or suppress this variation: nonzero dropout does not guarantee different answers or rewards in every group. Native attention-mask dimensions grow with the causal prefix. Within-chain sharing therefore refers to consistent decisions on overlapping positions, rather than equality of differently shaped tensors. This behavior was verified for the tested release configuration; backend dependence is discussed in Appendix C.1. The proofs require replay of the corresponding noise at each call and also cover schedules without temporal sharing. The policy is an average over execution noise; its optional Bayesian interpretation is not used in the derivations.

Mask replay. At update time we regenerate $\xi^{(k)}$ from the stored RNG state and recompute $\log \pi_\theta(y^{(k)} \mid x, \xi^{(k)})$ under the same execution noise used at the corresponding rollout calls. When generation and scoring implement the same conditional computation, this gives identical mathematical likelihoods at $\theta = \theta_{\mathrm{old}}$, as formalized in Proposition 6. Exact floating-point reproduction requires additional implementation conditions discussed in Appendix C.1.

B.2 Well-definedness of the latent gradient

Assumption 1. Fix a prompt x, a finite latent depth T, execution noise ξ, and a finite answer y with positive conditional probability. In a neighborhood of the evaluated computation, $G_\theta(x; \xi)$ is continuously differentiable in θ, and each $F_{t,\theta}(S; \xi)$ is jointly continuously differentiable in θ and the continuous state S. For every fixed answer prefix, the conditional token log-likelihood, including the

intervening answer-cache updates, is jointly continuously differentiable in θ and the initial decoding state S_T. Token identities, position indices, causal masks, and ξ are held fixed when taking these derivatives.

This is a local sufficient condition on the executed computation. Smooth activations (GELU or SiLU), softmax on the allowed attention positions, and LayerNorm or RMSNorm with a positive stabilizing constant satisfy it for finite inputs; fixed dropout masks act as linear maps on their inputs. The condition explicitly includes the prompt pass and cache dependencies.

Lemma 2 (Well-defined latent gradient)**.** *Under Assumption 1, the complete-state Jacobian $J_t := dS_t/d\theta \in \mathbb{R}^{d_t \times D}$ exists at the evaluated parameters for every $0 \le t \le T$ and obeys*

$$J_0 = \frac{\partial G_\theta(x;\xi)}{\partial \theta}, \qquad J_t = \left.\frac{\partial F_{t,\theta}(S;\xi)}{\partial \theta}\right|_{S=S_{t-1}} + \left.\frac{\partial F_{t,\theta}(S;\xi)}{\partial S}\right|_{S=S_{t-1}} J_{t-1}. \tag{6}$$

The latent Jacobian $dh_T/d\theta$ is the corresponding block of J_T, and $\nabla_\theta \log \pi_\theta(y \mid x, \xi)$ is well defined.

Proof in Appendix D.1. The partial parameter derivative in (6) holds the input state fixed; the second term propagates all earlier dependencies, including those through cached keys and values. Gradient-enabled replay computes this derivative provided the prompt, latent, and answer-cache computations remain connected to autograd. Reusing detached rollout caches would omit these paths. Fixed execution noise is treated as constant, without differentiating its sampling operation.

B.3 Sequence-sum identity and sequence-mean bias

To distinguish the theoretical estimator from the implementation, define the idealized policy-only, sequence-sum surrogate

$$\mathcal{L}^{\mathrm{surr}}_{\mathrm{sum}}(\theta) = \mathbb{E}_{x,\mathcal{G}}\left[\frac{1}{K}\sum_{k=1}^{K}\sum_{\ell} \min\Big(\rho^{(k)}_{\theta,\ell} A^{(k)}, \mathrm{clip}(\rho^{(k)}_{\theta,\ell}, 1\pm\varepsilon) A^{(k)}\Big)\right], \tag{7}$$

where groups are sampled from the fixed old policy as above. Unlike (3), this expression has no $1/|y^{(k)}|$ divisor and no regularization term.

Theorem 3 (Sequence-sum policy-gradient identity)**.** *Let $K \ge 2$, $\varepsilon > 0$, and let the rollouts be i.i.d. draws from $\Pi_{\theta_{\mathrm{old}}}(\cdot \mid x)$. Assume a normalized differentiable policy with locally fixed support, and sufficient integrability to interchange the relevant expectations and derivatives. Rewards are parameter-independent. Under mask replay and detached advantages $A^{(k)} = r^{(k)} - \mu_r$, the surrogate* (7) *satisfies*

$$\nabla_\theta \mathcal{L}^{\mathrm{surr}}_{\mathrm{sum}}(\theta)\Big|_{\theta=\theta_{\mathrm{old}}} = \left(1 - \tfrac{1}{K}\right) \nabla_\theta J(\theta)\Big|_{\theta_{\mathrm{old}}}. \tag{8}$$

Thus multiplying its sampled gradient by $K/(K-1)$ gives an unbiased estimator of $\nabla J(\theta_{\mathrm{old}})$.

Proof. See Appendix D. □

Corollary 4 (Sequence-sum target on the marginal policy)**.** *Under Theorem 3, the corrected sequence-sum estimator targets the gradient of the expected reward of the marginal policy $\bar{\pi}_\theta$ at θ_{old}. This follows by marginalizing the parameter-independent mask law in J. It does not imply unbiasedness of the sequence-mean update in Algorithm 1.*

Effect of sequence-mean aggregation. Let $L = |Y| \ge 1$, $s = \nabla_\theta \log \pi_\theta(Y \mid x, \xi)|_{\theta_{\mathrm{old}}}$, and $m_x = \mathbb{E}[r \mid x]$. Under the same sampling and regularity conditions, but retaining the length divisor, the policy-only estimator at θ_{old} is $\widehat{g}_{\mathrm{mean}} = K^{-1}\sum_k (r^{(k)} - \mu_r) s^{(k)}/L_k$. Its conditional expectation is

$$\mathbb{E}[\widehat{g}_{\mathrm{mean}} \mid x] = \left(1 - \tfrac{1}{K}\right)\left(\mathbb{E}\Big[\frac{rs}{L}\Big|x\Big] - m_x \mathbb{E}\Big[\frac{s}{L}\Big|x\Big]\right). \tag{9}$$

In general $\mathbb{E}[s/L \mid x] = \nabla_\theta \mathbb{E}_\theta[1/L \mid x]|_{\theta_{\mathrm{old}}} \neq 0$, so the baseline cross-terms do not vanish. Equivalently, the bracket is $\nabla_\theta \mathbb{E}_\theta[r/L \mid x] - m_x \nabla_\theta \mathbb{E}_\theta[1/L \mid x]$ evaluated at θ_{old}, not the gradient of $\mathbb{E}_\theta[r/L \mid x]$ alone. This characterizes the normalization bias of the policy-only estimator under the stated sampling law. Appendix D.3 derives the identity and gives a counterexample; Appendix C.3 explains its role in the implementation.

Remark 5 (Why σ_r is dropped). The original GRPO advantage $(r^{(k)} - \mu_r)/(\sigma_r + \delta)$ couples $s^{(k)}$ to every $r^{(j)}$ via a nonlinear function of all rewards; the factorization $\mathbb{E}[r^{(j)}s^{(k)}] = \mathbb{E}[r^{(j)}]\mathbb{E}[s^{(k)}]$ used above breaks and the estimator is biased. Liu et al. (2025b) report that this bias overweights low-variance prompts and underweights high-variance ones; we drop σ_r throughout.

B.4 Likelihood consistency under mask replay

Replay conditions scoring on the execution noise that produced the rollout's latent state. Here ξ includes every stochastic choice in the model computation that affects the conditional token probabilities; answer tokens and their prefixes are held fixed during scoring. For the shared-mask release schedule, this includes the within-chain pattern and its extension to new attention positions. More generally, the same tape and execution schedule must reproduce the corresponding masks between generation and replay, whether or not they are shared across steps.

Proposition 6 (Likelihood consistency under replay)**.** *Fix a prompt x, rollout noise ξ, and an answer y whose scored conditional token probabilities are positive under θ_{old}. Assume that generation and replay evaluate the same deterministic conditional computation given parameters, x, ξ, and each fixed answer prefix, including the prompt pass and cache updates. If the parameters remain θ_{old}, then the replayed token probabilities equal their rollout-time values, and*

$$\rho_{\theta_{\mathrm{old}},\ell} = \frac{\pi_{\theta_{\mathrm{old}}}(y_\ell \mid x, \xi, y_{<\ell})}{\pi_{\theta_{\mathrm{old}}}(y_\ell \mid x, \xi, y_{<\ell})} = 1. \tag{10}$$

For $\varepsilon > 0$, PPO clipping is therefore inactive at this evaluation point.

Proof. See Appendix D.4. □

Drawing a new mask and generating a new answer yields a new rollout. Attaching a new mask to an existing answer instead generally changes its conditional likelihood and need not give a ratio of one, even at the same parameters. Replay supplies consistency for the conditional ratios used here; it does not establish a reduction in gradient variance.

The policy-gradient identity of Theorem 3 is evaluated only at the generating parameters. For $\theta \neq \theta_{\mathrm{old}}$, scoring old-policy answers with $A\nabla_\theta \log \pi_\theta$ alone is generally not an unbiased estimator of $\nabla J(\theta)$. Replay does not remove this sampling-distribution mismatch. The sampled-answer implementation addresses the answer-sampling requirement; equality of replayed probabilities alone would not establish it. The theorem additionally requires the sampling distribution to match the scored policy.

C Detailed Algorithm and Regularization

This section describes the dropout-GRPO algorithm (Appendix C.1) and three implementation refinements used in practical training (Appendices C.2–C.4).

C.1 Algorithm

The procedure is summarized in Algorithm 1. For each prompt x, generate K rollouts with attention dropout active and sample the answer tokens from the conditional policy. During replay, hold those sampled token identities fixed. In the idealized sampling law, each rollout has an independent execution random tape $\xi^{(k)}$. In the implementation, CPU and CUDA RNG states are stored at rollout-chunk boundaries and restored for scoring with the same batch structure and call schedule. The noise covers prompt processing, latent recurrence, and answer decoding, rather than only the latent phase. Under the matched-computation assumptions of Proposition 6, replayed conditional likelihoods equal their rollout-time values at $\theta = \theta_{\mathrm{old}}$. Sequence-mean aggregation has the normalization bias characterized in Appendix B.3; it does not implement the idealized sequence-sum surrogate of Theorem 3.

Native attention dropout and replay schedule. The Qwen2.5 implementation applies the configured dropout probability inside attention, after softmax and before value aggregation, in each Transformer layer. No feature-mask injection is required. Both explicit attention and scaled-dot-product attention support this operation. The release procedure restores the chain RNG state before each single-token forward. In controlled tests using the release code with PyTorch 2.9.1, Transformers 4.46.2, BF16 CUDA SDPA, and six latent steps, mask decisions were identical on overlapping attention positions across steps within each rollout, while differing across rollouts. Tests covered multiple prefix lengths and seeds, including a 28-layer case; all 12,238,080 cross-step position comparisons agreed. These were random-weight implementation tests, not additional training experiments. As the prefix grows, new positions extend the mask. This is the shared-mask schedule illustrated in Figure 2.

This behavior depends on the attention backend and precision: eager attention and FP32 SDPA produced cross-step differences in the same audit. Thus RNG resetting alone is not a backend-independent shared-mask guarantee. The release's Python dropout auditor does not observe fused SDPA calls; its zero-check reports must not be interpreted as mask verification. The SDPA tests directly recovered the post-dropout attention coefficients and separately checked rollout-to-update replay.

The newer `native_per_step` implementation advances the dropout RNG normally and replays its full call sequence, with a separate RNG for categorical answer sampling. It is an alternative schedule covered by the general analysis, not the shared-mask release schedule described here. The reported training runs do not compare these schedules. Feature-locked dropout is also outside this paper's method.

The trainer accumulates gradients over rollout chunks and takes one optimizer step per rollout batch. Parameters are unchanged between the old-policy scoring pass and gradient-enabled replay; the chunks are not successive PPO mini-epochs. Under exact replay the policy ratios are one and clipping is inactive during this gradient evaluation. Replay is used to score each answer under its rollout noise, without a claimed multi-epoch variance-reduction benefit.

The proposition concerns exact arithmetic. Bit-identical floating-point reproduction additionally requires matching random-number consumption, tensor shapes, cache handling, and numerical kernels between generation and scoring. Restoring an RNG state alone does not establish these conditions; rollout/replay comparisons are an implementation check.

Algorithm 1 Dropout-GRPO for latent-reasoning LLMs

Require: $\pi_{\theta_{\text{old}}}$, $\mathcal{D}$, K, p, T, ε, β, η_t, $[\alpha_{\min}, \alpha_{\max}]$, δ.

1: **for** $t = 1, \ldots, T_{\max}$ **do**

2: Sample mini-batch $x \sim \mathcal{D}$. For $k = 1, \ldots, K$: record RNG state and execution schedule for $\xi^{(k)}$, unroll (4) with native attention dropout under the within-chain shared-mask schedule (Appendix C.1), sample answer $y^{(k)}$ from the conditional policy, record $\log \pi_{\theta_{\text{old}}}(y^{(k)} \mid x, \xi^{(k)})$, compute $r^{(k)}$.

3: Compute μ_r. If $\mu_r \notin [\alpha_{\min}, \alpha_{\max}]$, drop group (DDP-coordinated, §C.4); else set $A^{(k)} = r^{(k)} - \mu_r$.

4: For each kept k: replay the corresponding dropout calls using the stored RNG state and schedule, recompute $\log \pi_\theta(y^{(k)} \mid x, \xi^{(k)})$ and $\rho_\ell^{(k)}$.

5: Take a gradient step on $\mathcal{L}(\theta) = -\frac{1}{K_{\text{kept}}} \sum_k \frac{1}{|y^{(k)}|} \sum_\ell \min(\rho_\ell^{(k)} A^{(k)}, \text{clip}(\rho_\ell^{(k)}, 1 \pm \varepsilon) A^{(k)}) + \beta \frac{1}{K_{\text{kept}}} \sum_k \frac{1}{|y^{(k)}|} \sum_\ell \tilde{k}_3(r_\ell^{\text{ref}})$, with fixed β, $r_\ell^{\text{ref}} = \log \pi_{\text{ref}} - \log \pi_\theta$ at the scored token, and $\tilde{k}_3$ from (12); set $\theta_{\text{old}} \leftarrow \theta$.

6: **end for**

C.2 Correctly oriented k_3 and Huber regularization

For a fixed decoding context, write $p = \pi_\theta$ and $q = \pi_{\text{ref}}$ on a common positive support. The nonnegative k_3 statistic (Schulman, 2020) uses the reference-to-policy log-ratio

$$r^{\text{ref}} := \log q - \log p, \qquad k_3(r^{\text{ref}}) = e^{r^{\text{ref}}} - r^{\text{ref}} - 1, \qquad \mathbb{E}_{Y \sim p}[k_3(r^{\text{ref}}(Y))] = \text{KL}(p \| q). \quad (11)$$

The expectation identity requires samples from p. The opposite log-ratio with the same exponential formula does not estimate this KL; Appendix D.5 derives both expectations.

The corrected trainer offers either the full statistic or a Huber-style tangent extension. Writing $u = r^{\mathrm{ref}}$, the latter is

$$\tilde{k}_3(u) = \begin{cases} e^u - u - 1 & |u| \le \delta, \\ (e^\delta - 1)(u - \delta) + (e^\delta - \delta - 1) & u > \delta, \\ (e^{-\delta} - 1)(u + \delta) + (e^{-\delta} + \delta - 1) & u < -\delta. \end{cases} \tag{12}$$

This extension is nonnegative and C^1, with derivative in $[e^{-\delta} - 1, e^\delta - 1]$ and nonzero slopes in both tails. It avoids the zero outside-gradient of clamping the log-ratio before applying k_3. Only the scalar derivative is bounded: the penalty value remains unbounded, and the model-parameter gradient also depends on the log-probability Jacobian. We use $\delta = 5$, whose maximum scalar slope is $e^5 - 1 \approx 147$.

Huberization changes the regularizer. Convexity gives $0 \le \tilde{k}_3(u) \le k_3(u)$, with strict upper inequality in the tails, so it is not an unbiased KL statistic when a supported outcome enters those tails. The code evaluates the central expression with `expm1` on a bounded argument and uses linear tails. Setting `use_huberized_kl=False` selects full, unclamped k_3.

In Equation (3), $\mathcal{R}_{\theta_{\mathrm{old}}}(\theta)$ denotes the expectation, over the stated old-policy groups, of $K^{-1} \sum_k |y^{(k)}|^{-1} \sum_\ell H(r_{\theta,\ell}^{\mathrm{ref},(k)})$, where $H = \tilde{k}_3$ in Algorithm 1, or $H = k_3$ for the full-statistic option. The reference-to-policy ratio is evaluated at each fixed answer prefix with the specified reference computation. The trainer differentiates this frozen-sample penalty directly. Even without Huberization, an unbiased KL *value* statistic does not by itself provide an unbiased gradient of the current-policy KL: the derivative of that expectation includes its sampling-distribution term (Appendix D.5). Theorem 3 concerns only the reward surrogate and supplies no guarantee for this regularizer.

Correction and experimental scope. The original implementation and manuscript used the opposite log-ratio. The expressions above document the corrected implementation. The reported experimental results are retained from the original runs and are not a reevaluation of the corrected regularizer.

C.3 Sequence-mean loss aggregation

In all reported experiments, we average the per-token loss within each rollout and then average across kept rollouts, as in Algorithm 1. This normalizes the token sum by the realized response length, so a longer answer does not receive a larger loss solely from containing more terms of the same magnitude. It is a practical aggregation choice, not an unbiased estimator of the expected sequence-reward gradient. Because completion length depends on the sampled answer, the resulting gradient may change direction as well as scale.

Equation (9) isolates this effect for unfiltered on-policy groups; Appendix D.3 gives an explicit counterexample. Theorem 3 applies only to the idealized sequence-sum policy surrogate. We retain sequence-mean aggregation in the experimental algorithm and make no general variance-reduction guarantee for this normalization. Its bias and any empirical stability benefit must be distinguished from the sequence-sum identity.

C.4 Computational group filtering

Motivated by the zero-advantage groups identified in the dynamic-sampling diagnostic of Yu et al. (2025), we use an accuracy filter to save computation. After rollout generation, groups whose empirical accuracy μ_r lies outside $[\alpha_{\min}, \alpha_{\max}]$ are skipped before subsequent scoring and backpropagation. Default values are $\alpha_{\min} = 0.20$ and $\alpha_{\max} = 0.95$. This is an implementation heuristic; its computational motivation and scope are discussed in Section 5.

D PROOFS

D.1 PROOF OF LEMMA 2 (WELL-DEFINED LATENT GRADIENT)

Fix x, ξ, a finite depth T, and a finite answer y as in Assumption 1. All discrete execution choices are fixed. We induct over the complete continuous state, not only the latest latent.

Base case ($t = 0$). The prompt computation $S_0 = G_\theta(x;\xi)$ is continuously differentiable by assumption. Its derivative $J_0 = \partial_\theta G_\theta(x;\xi)$ includes both the initial hidden vector and all prompt-cache entries.

Inductive step. Suppose $S_{t-1}(\theta)$ is continuously differentiable locally. Joint continuous differentiability of $F_{t,\theta}(S;\xi)$ and the multivariate chain rule give

$$\frac{dS_t}{d\theta} = \left.\frac{\partial F_{t,\theta}(S;\xi)}{\partial \theta}\right|_{S=S_{t-1}} + \left.\frac{\partial F_{t,\theta}(S;\xi)}{\partial S}\right|_{S=S_{t-1}} \frac{dS_{t-1}}{d\theta}.$$

The state derivative is a $d_t \times d_{t-1}$ matrix, so the identity also covers growing caches. Its columns include dependencies on the latest hidden vector and every cached prefix key and value. Since the unroll is finite, the local composition yields (6) through T; projecting onto the hidden-vector coordinates gives $dh_T/d\theta$.

For the fixed answer prefix $y_{<\ell}$, set $a_\ell(\theta, S) = \log \pi_\theta(y_\ell \mid x, S, y_{<\ell}, \xi)$, where decoding from S includes all subsequent answer-cache updates. The remaining part of Assumption 1 yields

$$\nabla_\theta \log \pi_\theta(y \mid x, \xi) = \sum_{\ell=1}^{|y|} \left[\nabla_\theta a_\ell(\theta, S_T) + J_T^\top \nabla_S a_\ell(\theta, S_T)\right],$$

where both gradients on the right are partial derivatives holding the other argument fixed. Thus the conditional sequence-score gradient exists. No derivative through sampled token identities or the fixed noise is needed. □

D.2 PROOF OF THEOREM 3 (SEQUENCE-SUM POLICY-GRADIENT IDENTITY)

At $\theta = \theta_{\text{old}}$, mask replay yields $\rho^{(k)}_{\theta_{\text{old}},\ell} \equiv 1$ for every token, so the PPO clip is inactive. Differentiating the sequence-sum surrogate (7), with advantages held fixed, sums the token scores without a length divisor. Writing $s^{(k)} := \nabla_\theta \log \pi_\theta(y^{(k)} \mid x, \xi^{(k)})|_{\theta_{\text{old}}}$,

$$\nabla_\theta \mathcal{L}^{\text{surr}}_{\text{sum}}\big|_{\theta_{\text{old}}} \;=\; \mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K}(r^{(k)} - \mu_r)\, s^{(k)}\right].$$

Fix x; all expectations in the following calculation are conditional on this prompt. Normalization and the differentiation assumptions give $\mathbb{E}[s^{(k)} \mid x] = 0$. Expanding the baseline term and using i.i.d. rollouts,

$$\mathbb{E}[\mu_r\, s^{(k)}] \;=\; \frac{1}{K}\sum_{j=1}^{K}\mathbb{E}[r^{(j)}\, s^{(k)}] \;=\; \frac{1}{K}\mathbb{E}[r^{(k)}\, s^{(k)}],$$

because for $j \neq k$ the independence factorization combined with the standard zero-mean property of the score function under $\Pi_{\theta_{\text{old}}}$ gives $\mathbb{E}[r^{(j)}\, s^{(k)}] = \mathbb{E}[r^{(j)}]\,\mathbb{E}[s^{(k)}] = 0$. Substituting,

$$\mathbb{E}[(r^{(k)} - \mu_r)\, s^{(k)}] \;=\; \left(1 - \tfrac{1}{K}\right)\mathbb{E}[r^{(k)}\, s^{(k)}].$$

Since $p(\xi)$ is parameter-independent, $s^{(k)}$ is also the joint score of Π_θ. Thus $\mathbb{E}[r^{(k)} s^{(k)} \mid x] = \nabla_\theta \mathbb{E}_{\Pi_\theta}[r \mid x]|_{\theta_{\text{old}}}$. Averaging over k and then x yields (8). □

D.3 SEQUENCE-MEAN NORMALIZATION BIAS

Under the sampling assumptions of Theorem 3, fix a prompt x and let $u_k = s^{(k)}/L_k$. All expectations below condition on x. For $j \neq k$, independence gives $\mathbb{E}[r^{(j)}u_k] = m_x\mathbb{E}[u_k]$, but $\mathbb{E}[u_k]$ need not be zero. Consequently,

$$\mathbb{E}[\mu_r u_k] = \frac{1}{K}\mathbb{E}[r^{(k)}u_k] + \frac{K-1}{K}m_x\mathbb{E}[u_k], \qquad \mathbb{E}[(r^{(k)} - \mu_r)u_k] = \frac{K-1}{K}\big(\mathbb{E}[ru] - m_x\mathbb{E}[u]\big).$$

Averaging over k proves (9). For each fixed answer, $L(Y)$ is a fixed statistic; differentiating the probability of that answer gives $\mathbb{E}[s/L] = \nabla_\theta \mathbb{E}_\theta[1/L]|_{\theta_{\text{old}}}$ and $\mathbb{E}[rs/L] = \nabla_\theta \mathbb{E}_\theta[r/L]|_{\theta_{\text{old}}}$. No derivative of a sampled discrete length is taken.

For a counterexample, consider two answers with success probability $p = \text{sigmoid}(\theta)$. Success has reward one and length one; failure has reward zero and length two, with a deterministic second token. At $p = 1/2$, their sequence scores are $1/2$ and $-1/2$, respectively, and $\nabla J = 1/4$. For $K = 2$, homogeneous groups have zero advantages. Each mixed group has sequence-mean estimator $\frac{1}{2}[(\frac{1}{2})(\frac{1}{2}) + (-\frac{1}{2})(-\frac{1}{2})/2] = 3/16$, and mixed groups occur with probability $1/2$. Hence $\mathbb{E}[\widehat{g}_{\text{mean}}] = 3/32$, whereas $(1 - 1/K)\nabla J = 1/8$. The sequence-sum estimator does give $1/8$. Even if all answers have a common fixed length L_0, sequence-mean aggregation introduces an extra factor $1/L_0$; if fixed lengths vary across prompts, it reweights prompt gradients. Multiplication by $K/(K-1)$ corrects self-inclusion but not the variable-length bias.

D.4 Proof of Proposition 6 (likelihood consistency)

Fix x, ξ, y, and $\theta = \theta_{\text{old}}$. The prompt computation has the same inputs and execution noise in generation and replay, so its complete state is identical in exact arithmetic. Inductively, each latent transition receives the same prior state, parameters, and noise for that call, obtained from the same random tape and execution schedule. Its updated hidden vector and cache therefore coincide. This induction compares corresponding calls between generation and replay. It applies to the shared-mask schedule and, more generally, does not require temporal sharing within a rollout. The same argument applies to answer decoding: at each position, replay uses the fixed answer prefix and the same conditional computation, including its cache updates and any probability transformations. Thus its conditional token probabilities equal those evaluated during generation. Positivity makes the ratio well defined, and equality gives (10). Since 1 lies strictly inside $[1 - \varepsilon, 1 + \varepsilon]$ for $\varepsilon > 0$, clipping is inactive there. This is a conditional-likelihood identity; it makes no assertion about estimator variance or unbiasedness away from θ_{old}. □

D.5 KL orientation and the sampled regularizer

Fix a decoding context and normalized distributions $p = p_\theta$ and q on a common finite positive support, with q fixed. Let $u(y) = \log(q(y)/p(y))$ and $s(y) = \nabla_\theta \log p(y)$. Direct summation yields

$$\mathbb{E}_p[e^u - u - 1] = \sum_y q(y) + \sum_y p(y) \log \frac{p(y)}{q(y)} - 1 = \text{KL}(p\|q).$$

In contrast, with $c = -u = \log(p/q)$, the historical convention gives

$$\mathbb{E}_p[e^c - c - 1] = \sum_y \frac{p(y)^2}{q(y)} - 1 - \text{KL}(p\|q),$$

which is not generally $\text{KL}(p\|q)$. For example, $p = (0.8, 0.2)$ and $q = (0.5, 0.5)$ give approximately 0.167255 for the historical statistic, whereas the KL and the corrected statistic's expectation are 0.192745.

The tangent extension in (12) matches both value and slope at $\pm\delta$. The exponential is strictly convex, so each tail tangent lies strictly below k_3 away from its matching point. The outward tail slopes increase the value from nonnegative boundary values. Hence $0 \le \tilde{k}_3 \le k_3$, strictly below it in either tail. Its expectation is therefore strictly below KL whenever an outcome with positive probability lies in a tail.

Finally, for any continuously differentiable scalar penalty H, $\nabla_\theta u = -s$ and the product rule give

$$\nabla_\theta \mathbb{E}_{p_\theta}[H(u)] = \mathbb{E}_{p_\theta}[(H(u) - H'(u))s]. \tag{13}$$

For $H = k_3$, $H - H' = -u$, yielding the true KL gradient $\mathbb{E}_p[\log(p/q)s]$. Holding samples fixed and differentiating only $H(u)$ instead gives $-H'(u)s$, omitting the sampling term $H(u)s$. Thus the corrected sign establishes the value identity, not a full KL policy-gradient guarantee for direct backpropagation through the sampled penalty. These statements concern a fixed context and current-policy samples; sequence-mean aggregation and old-policy sampling in the practical surrogate do not inherit the value or gradient identities without further analysis.

F Ablation Studies

All results in this appendix were rerun with GSM-Hard and SVAMP excluded from the GRPO training set, matching the revised proposed-method setup in Section 4.2.

F.1 Component ablations

Table 4 compares the proposed method with variants that change the dropout schedule, remove mask replay, replace the Huberized k_3 penalty with a hard clamp, or remove the group-level accuracy filter. The MC-dropout variant resamples masks across latent steps rather than sharing the within-chain pattern. The no-replay variant instead draws fresh masks when scoring the generated answers for the policy update.

Table 4: GSM8K pass@1 accuracy for the proposed method and its component ablations. Boldface marks the highest reported mean.

Variant	Accuracy
Dropout-GRPO (proposed)	$\mathbf{29.39\% \pm 0.16\%}$
MC dropout	$29.21\% \pm 0.27\%$
Fresh masks at policy update (no mask replay)	$28.02\% \pm 0.49\%$
Without Huberized k_3 KL (hard clamp instead)	$28.91\% \pm 0.42\%$
Without group-level accuracy filter	$28.99\% \pm 0.66\%$

While the proposed method has the highest reported mean accuracy, we found that the MC dropout method worked comparably well. Removing replay produces the largest reduction in accuracy, which is consistent with the role of replay in preserving the computation used to generate each answer.

While the huberized KL and group-level accuracy filter may not seem like a big factor in the end accuracy, their main purpose was for stabilization in the training process, which can be seen in their greater variance of resulting accuracy.

F.2 Evaluation with and without dropout

Table 5 compares the SFT baseline and the trained model with dropout enabled and disabled at inference. Training improves accuracy under both settings, by 2.84 percentage points with dropout on and 2.10 points with dropout off.

Table 5: GSM8K evaluation under the two inference settings. Improvements are absolute percentage-point differences between the reported means.

Model	Dropout ON (marginal policy, J)	Dropout OFF (deterministic)
Coconut SFT (baseline)	21.46%	27.29%
Coconut + Dropout-GRPO	$24.30\% \pm 0.40\%$	$29.39\% \pm 0.16\%$
Improvement from training	$+2.84$ pp	$+2.10$ pp

The dropout-ON column is a Monte Carlo estimate of $J(\theta)$, the expected reward of the mask-marginalized policy. Training raises this estimate from 21.46% to 24.30%, a gain of 2.84 percentage points, alongside the 2.10-point gain under deterministic deployment. Thus, the construction measurably improves the objective it is designed to optimize. The deterministic gain is approximately 74% of the dropout-ON gain ($2.10/2.84$), indicating that roughly three-quarters of the improvement carries over to deployment without dropout. This compares aggregate accuracy gains; it does not establish an unbiased gradient for the implemented training update.