A CRITICAL LOOK AT POWER SAMPLING WHEN TEST-TIME SHARPENING LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) has substantially improved LLM reasoning, yet growing evidence suggests it mainly performs *distribution sharpening*: concentrating probability mass on reasoning paths the base model already produces. Recent work therefore seeks RLVR's gains without needing reward signals, by sampling from the power distribution $\pi_\alpha \propto \pi_{\text{base}}^{\alpha}$ via Markov chain Monte Carlo or sequential Monte Carlo. We show that these samplers fail to converge to π_α under realistic compute budgets, and that their gains are largely recreated via simple Best-of-N with appropriate grading . This suggests they succeed by selecting high-probability sequences, not by approximating π_α, raising a natural question: is the power distribution itself a worthwhile target? We test this hypothesis directly by fine-tuning the base model to match π_α, sidestepping sampler convergence issues entirely. Despite approximating the target far more faithfully than existing samplers, this strategy yields gains that are noisy and specific to certain models and sampling tasks. Together, our results suggest that the value of power sampling lies not in faithfully targeting π_α, but in the simpler goal of extracting sequences the base model already deems likely.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become a cornerstone of large language model (LLM) post-training (Shao et al., 2024; Guo et al., 2025; Hu et al., 2025; Lambert et al., 2025). It has proven especially effective in domains with checkable answers, such as mathematics, code generation, and formal reasoning, where it elicits multi-step reasoning that is difficult to instil through supervised fine-tuning alone (Chen et al., 2021; Li et al., 2022; Hou et al., 2024; Zimmer et al., 2025).

A growing body of work, however, questions whether RLVR unlocks genuinely new reasoning behaviour or merely amplifies what the base model can already do. Yue et al. (2025) find that its accuracy gains come at the cost of diversity, as revealed by pass@k, while He et al. (2025) show that GRPO exhibits a degenerate rank bias, funnelling additional probability mass toward sequences the base model already ranks highly. These findings raise a natural question: can the gains attributed to RLVR be recovered simply by sampling more cleverly from the base model's own high-probability regions, with no verifier or reward signal at all?

A growing body of work claims the answer is yes: simply sample from the *power distribution* $\pi_\alpha \propto \pi_{\text{base}}^{\alpha}$ with $\alpha > 1$.Karan & Du (2025) proposed Markov chain Monte Carlo (MCMC) sampling, and subsequent work turned to sequential Monte Carlo (SMC) for efficiency (Azizi et al., 2026; WANG et al., 2026). These methods are motivated by a compelling theory — that faithfully approximating π_α should yield substantial reasoning improvements.

In this work, we critically examine this popular claim. We analyze why existing samplers fail, propose POWER-RLOO to directly target π_α with high fidelity, and test whether faithful approximation actually drives reasoning gains. Our contribtuions and conclusions are as follows:

- **We show that recent SMC schemes fail to converge** to π_α (Section 3), instead behaving as likelihood-guided search with little sensitivity to α (Figure 1).

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.

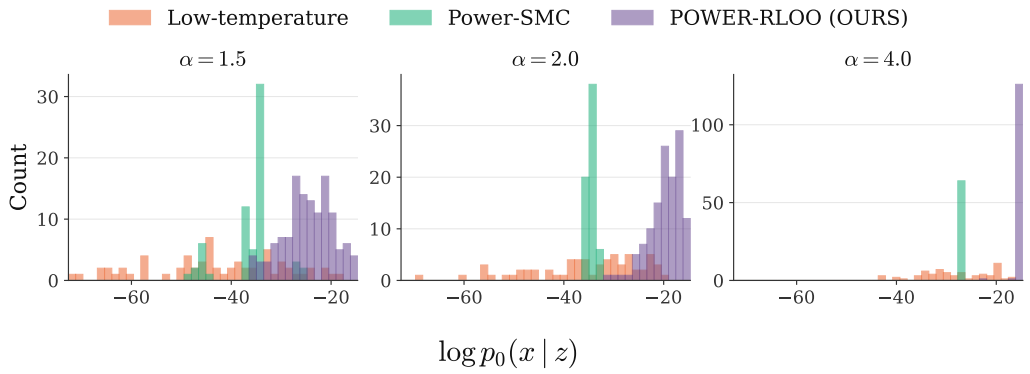


Figure 1: Base-model log-likelihoods of samples drawn from low-temperature decoding ($\tau = 1/\alpha$), Power-SMC, and our POWER-RLOO on Qwen2.5-7B-Instruct (MATH500, Question 1), for $\alpha \in \{1.5, 2.0, 4.0\}$. Power-SMC's samples collapse into a narrow, near-identical likelihood band that is largely insensitive to α, consistent with particle degeneracy. The narrow band indicates fewer than 10 effective samples. Power-RLOO, in contrast, shifts its mass toward higher likelihood as α grows, matching the behavior expected under π_α.

- **We find that SMC gains are largely recreated by a simple Best-of-N variant** that periodically resamples the highest-likelihood partial sequences, suggesting that these samplers succeed by selecting high-probability sequences rather than by approximating π_α.
- **We introduce Power-RLOO as a fine-tuning scheme that faithfully targets** π_α. Critical to efficiency, we exploit the connection between power distribution and low-temperature sampling, allowing our adapter to learn only the non-local lookahead correction (Section 4).
- **We find that faithful approximation is insufficient.** Despite approximating the target far more effectively than previous work (Figure 1), we find that explicitly targeting π_α yields accuracy gains beyond low-temperature decoding only when the base model is already reasonably well-aligned with the task, contradicting claims that approximation quality drives improvements.
- **We show sharpening elicits latent capacity, not just selection.** Despite the inherently sharpening nature of the power distribution, modest fine-tuning actually increases pass@k, suggesting it reveals existing reasoning capacity rather than merely narrowing the output distribution—a capacity simpler selection methods can then also access. (Section 5.2).

2 Background and Related Work

Conditioned on a prompt z, a large language model (LLM) defines a probability distribution over the set of finite sequences via a series of conditional probabilities as follows. Let $\mathcal{V}$ denote the vocabulary (token set), and let $x = (x_1, x_2, \ldots, x_T) \in \mathcal{V}^T$ denote a generated sequence of length T, where $x_T = \texttt{<eos>}$ is a distinguished end-of-sequence token. The model defines a conditional distribution over the next token given the prompt and previously generated tokens:

$$p_0(x_t \mid z, x_{<t}),$$

where $x_{<t} = (x_1, \ldots, x_{t-1})$ denotes the (possibly empty) prefix of tokens generated so far. The probability of an entire sequence x given the prompt z is then obtained via the chain rule of probability:

$$p_0(x \mid z) = \prod_{t=1}^{T} p_0(x_t \mid z, x_{<t}). \tag{1}$$

This autoregressive factorization allows the model to generate sequences token-by-token, sampling each x_t from $p_\theta(\cdot \mid z, x_{<t})$ conditioned on all previously generated tokens, until the <eos> token is produced or a maximum length is reached.

2.1 SHARPENING LLMS AT TEST-TIME

Low Temperature Sampling. A common simple strategy for biasing generation towards higher-likelihood continuations is low temperature sampling, in which each conditional is rescaled by a parameter $\tau > 0$:

$$p_0^\tau(x_t \mid z, x_{<t}) \propto p_0(x_t \mid z, x_{<t})^{1/\tau}. \tag{2}$$

Taking $\tau < 1$ sharpens the distribution over the next token, concentrating probability mass on the model's most likely continuations at each step, with $\tau \to 0$ recovering greedy (most-likely) decoding and $\tau = 1$ recovering the original conditional $p_0(x_t \mid z, x_{<t})$. Sampling a full sequence under temperature τ then proceeds as above by applying the chain rule to these tempered conditionals, adding negligible additional cost over standard LLM sampling. However, as the rescaling in (2) is applied independently at each timestep, this sequence-level distribution is only ever tempering the model's *immediate* next-token beliefs — it has no mechanism for taking into account how a given choice of y_t shapes the likelihood of the tokens that follow it.

Power Sampling. To account for the likelihood of these "future paths," Karan & Du (2025) instead propose sampling from the *power distribution* which acts on the sequence as a whole:

$$\pi_\alpha(x \mid z) = \frac{p_0(x \mid z)^\alpha}{Z_\alpha(z)} \quad \text{for} \quad Z_\alpha(z) = \sum_{x' \in \mathcal{X}} p_0(x' \mid z)^\alpha, \qquad \alpha > 1. \tag{3}$$

Here, the normalizing constant $Z_\alpha(z)$, which sums over all possible sequences $x' \in \mathcal{X}$, is analytically intractable which means that direct simulation from (3) is infeasible as we no longer have an autoregressive structure. Although strategies have been proposed via approximate sampling algorithms, based on MCMC (Karan & Du, 2025) and SMC (Ji et al., 2026; Azizi et al., 2026). However, we will later demonstrate, in Section 3, that these samplers fail to accurately sample the power distribution (3) under realistic compute budgets.

2.2 FINE-TUNING LLMS

In Section 4 we will propose an alternative approach for sampling an LLM's power distribution — fine-tuning the LLM to directly target the power distribution.We now provide requisite overview of LLM fine-tuning methods.

Fine-tuning via reinforcement learning. RL fine-tuning algorithms such as GRPO (Shao et al., 2024) assign a reward $r(x)$ to each sample x drawn from the model and update the policy to shift probability mass toward high-reward samples. When this reward can be computed automatically by checking the output against a ground-truth answer or test suite, rather than by a learned reward model or human judgment, the setting is known as reinforcement learning with verifiable rewards (RLVR). Fine-tuning is achieved by optimizing a RL objective of the form

$$J_{\text{KL-RL}}(\theta) = \mathbb{E}_{x \sim \pi_\theta}\left[r(x)\right] - \beta\, D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{base}})\,, \tag{4}$$

where β controls the regularizing penalty for diverging from the base model. In maximizing (4), the fine-tuned model assigns more probability to high-reward trajectories. More precisely, as derived by Korbak et al. (2022), the fine-tuning update is a form of variational inference with target posterior

$$\pi^*_{\text{KL-RL}}(x) = \frac{1}{Z}\,\pi_{\text{base}}(x)\exp(r(x)/\beta)\,. \tag{5}$$

In practice, to implement this we use LoRA (Hu et al., 2021), a parameter-efficient fine-tuning paradigm that exploits the rank deficiency of LLMs, representing weight updates as low-rank matrices while keeping the base model π_0 frozen.

Self-Improvement. Our setting of fine-tuning towards the power distribution fits into the field of *self-improvement* (Huang et al., 2022; Wang et al., 2023; Yuan et al., 2025), in which intrinsic signals from within the model are used in place of external rewards or data. For example, Huang et al. (2024) explore using the model's own log-probability as a reward signal,

$$r_{\text{self}}(x \mid z) := \log \pi_{\text{base}}(x \mid z), \tag{6}$$

both at inference time and within an RLHF loop, to target a distribution over responses that maximizes the self-reward;

$$\hat{\pi}(z) \approx \arg\max_{x \in \mathcal{X}} r_{\text{self}}(x \mid z; \pi_{\text{base}}). \tag{7}$$

Notably, this target coincides with taking $\alpha \to \infty$ in the power distribution (3), recovering the mode of π_{base}. Huang et al. (2024) find that this reward signal consistently improves accuracy over greedy decoding across a range of benchmarks, lending empirical support to this choice of reward.

(Un)importance of reward quality. The underlying mechanics of RLVR remain an active area of research, and a growing body of work has begun to challenge some of its assumptions. Shao et al. (2026) show that mathematical reasoning can be improved even when trained on reward signals that are weakly, or even negatively, correlated with true correctness, while 1-shot RLVR (Wang et al., 2025) shows that a single training example can recover most of the gains typically attributed to training on thousands. In a related vein, self-certainty and other internal-feedback methods (Zhao et al., 2026) dispense with ground-truth answers and verifiers altogether, using the model's own confidence as the sole reward signal and still matching RLVR performance in-domain, suggesting that the specific correctness signal driving RLVR may matter far less than is commonly assumed.

3 The Pitfalls of Existing Sharpening Schemes

We consider three inference-time schemes recently proposed for sampling from π_α: Metropolis–Hastings (Karan & Du, 2025), rollout estimation of the power conditional (Ji et al., 2026), and sequential Monte Carlo (Azizi et al., 2026). Although each is theoretically exact in the infinite limit, we show that at practical budgets each scheme collapses to the same behavior: keeping the highest-likelihood candidates and discarding the rest. The samplers therefore act as a likelihood-guided search, largely insensitive to α, and their gains cannot be attributed to the power distribution.

MCMC. Karan & Du (2025) propose sampling from π_α approximately using a Metropolis–Hastings (MH) scheme. Candidate blocks, or subsequences, of tokens are iteratively proposed and accepted according to the MH acceptance ratio (8). The authors show that, compared to sampling from the base model (1), their strategy steers generation towards higher-probability sequences. However, the size of the underlying sequence space means that MCMC schemes of this kind suffer from slow mixing and substantial compute overhead per generated sample. Furthermore, proposing at the level of blocks rather than tokens means that the log-likelihoods being compared differ by tens of nats, which reduces the sensitivity to α.

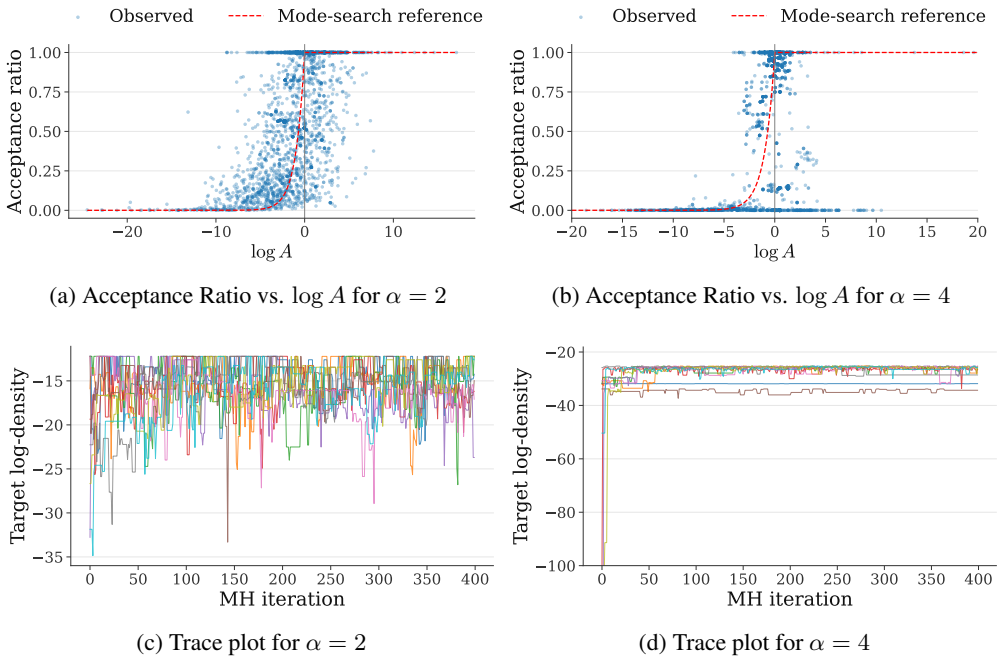
Let p_{prop} be the proposal that simulates new tokens x', conditional on the existing tokens x, which are assumed to be distributed according to π_α. The new tokens x' are accepted with probability

$$Acc(x', x) \;=\; \min\left(1,\; \underbrace{\frac{\pi_\alpha(x')}{\pi_\alpha(x)}}_{A} \cdot \underbrace{\frac{p_{\text{prop}}(x \mid x')}{p_{\text{prop}}(x' \mid x)}}_{B}\right). \tag{8}$$

Figure 2(a–b) shows that by increasing α, the term A begins to dominate the acceptance ratio (8). This means that a proposal is accepted whenever its log-probability is high, with the proposal term B contributing little to the probability of acceptance. We compare 400 MCMC steps against a greedy hill climber, which corresponds with $B = 1$ (red). The log-probability traces in Figure 2(c–d) show the same pattern: as α grows the chains stagnate at high-density regions rather than exploring. Despite this, Karan & Du (2025) demonstrate improved reasoning, suggesting that the benefit comes not from convergence to the power distribution but from the mode-seeking nature of their algorithm.

Rollout estimation of the twist. Ji et al. (2026) observe that the conditional distribution induced by π_α at each timestep is a tempered next-token distribution reweighted by a correction factor ζ_t,

$$\pi_\alpha(x_t \mid z, x_{<t}) = \frac{p_0(x_t \mid z, x_{<t})^\alpha\, \zeta_t(x_t)}{\sum_{x'_t \in \mathcal{V}} p_0(x'_t \mid z, x_{<t})^\alpha\, \zeta_t(x'_t)};\ \text{for } \zeta_t(x'_t) = \sum_{x_{t+1:T}} p_0(x_{t+1:T} \mid z, x_{<t}, x'_t)^\alpha, \tag{9}$$



(a) Acceptance Ratio vs. $\log A$ for $\alpha = 2$

(b) Acceptance Ratio vs. $\log A$ for $\alpha = 4$

(c) Trace plot for $\alpha = 2$

(d) Trace plot for $\alpha = 4$

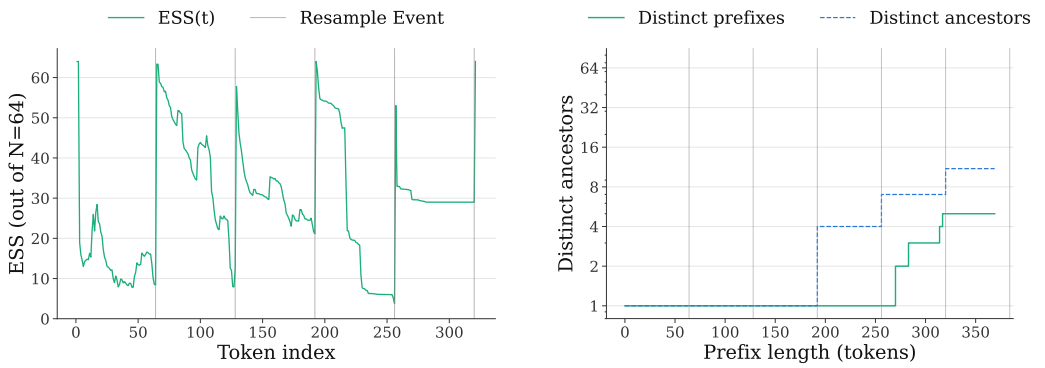
Figure 2: Diagnostics for the MCMC sampler. As α increases, the chains progressively fail to explore: the trace plots (c–d) show increasing stagnation, and the acceptance behavior (a–b) reveals the sampler collapsing into a greedy hill-climber that accepts almost exclusively on the basis of higher likelihood.

where $\mathcal{V}$ is the vocabulary. We refer to ζ_t, the tempered partition function over all completions, as the *twist*. Ji et al. (2026) estimate this intractable quantity using Monte Carlo with rollout blocks of horizon H from the base model. However, each term in this average is the likelihood of a completion hundreds of tokens long, raised to the power $\alpha - 1$, and such terms vary across rollouts by hundreds of nats. The average is therefore dominated by its largest term, so $\hat{\zeta}_t$ is effectively determined by the single best rollout. The candidate weights are then a softmax over the log-likelihood, so for any $\alpha > 1$ the selected block is approximately the one whose best rollout has the highest likelihood, motivating subsequent work that avoids explicit twist estimation altogether.

Power-SMC. Azizi et al. (2026) avoid estimating the twist altogether by maintaining a population of sequences each with importance weights. They run a standard SMC sampler on the *prefix targets* $\gamma_t(x_{1:t}) = p(x_{1:t} \mid z)^\alpha$. A population of N particles is advanced one token at a time under a prefix-only proposal $q_t(\cdot \mid z, x_{<t})$, each particle's weight is multiplied by the incremental importance weight $\omega_t = p_0(x_t \mid z, x_{<t})^\alpha / q_t(x_t \mid z, x_{<t})$, and the population is resampled whenever the effective sample size (ESS) falls below κN. However, the log-weight of a particle becomes a sum of t non-positive terms whose magnitudes vary from prefix to prefix, so the variance of log-weights across the population grows linearly in t and the ESS collapses within a few tokens of every reset.

Figure 3 shows the ESS trajectory at $\alpha = 4$ with $N = 64$ and $\kappa = 0.5$, where the ESS is checked every 64 tokens. Between resampling events, the ESS decays from 64 to ≈ 8 within 64 tokens (panel (a)), so that by the time of resampling the normalized weight is carried by a handful of particles. Panel (b) traces the ancestry of the final population: by the end of generation the particles share only 6 distinct prefixes. Combined with a proposal that is already sharp, the population from which the final sample is drawn consists of a few near-identical sequences, as demonstrated by the narrow, α-insensitive band of log-likelihoods in Figure 1.

To investigate this degeneracy further, we propose running a simple baseline where we replace weighted resampling by deterministic selection. Our *Best-of-N with Resampling* baseline runs N low-temperature samples in parallel and, every R tokens, keeps the k prefixes with the highest base-model log-likelihood and duplicates them to restore the population. Note that this simple selection



(a) ESS when resampling

(b) Distinct Ancestors

Figure 3: Ancestral diversity and effective sample size over the SMC trajectory. Resampling occurs whenever diversity drops too low, which, when compounded over the sequence, leaves the final population with few distinct ancestors and limited diversity.

rule ranks by cumulative log-likelihood whereas Power-SMC ranks by cumulative Rényi entropy of the traversed prefixes.

4 POWER-RLOO

We now propose Power-RLOO, a method to fine-tune an LLM to directly target π_α, avoiding the need for SMC approximations. We establish that π_α is the optimum of a KL-regularized RL objective with log-likelihood rewards, and derive an efficient token-level gradient estimator. This advance will enable a direct analysis of whether approximating π_α improves reasoning. Throughout, z denotes the prompt, $x = (x_1, \dots, x_T)$ a sampled response, $\pi_0 \equiv \pi_{\text{base}}$ the frozen base model, and π_θ the model being fine-tuned, initialized at π_0.

Sharpening as KL-regularized RL. Recall from (5) that the KL-regularized fine-tuning objective in (4) is maximized by $\pi^*(x \mid z) \propto \pi_0(x \mid z) \exp\big(r(x)/\beta\big)$. Choosing the reward

$$r(x) = \beta(\alpha - 1) \log \pi_0(x \mid z) \tag{10}$$

gives $\pi^* \propto \pi_0^\alpha$, so recovers the power distribution π_α. Substituting this reward into (4) gives:

$$J_{\text{KL-RL}}(\theta) = \beta\, \mathbb{E}_{x \sim \pi_\theta(\cdot|z)}\big[\alpha \log \pi_0(x \mid z) - \log \pi_\theta(x \mid z)\big] = -\beta\Big[D_{\text{KL}}\big(\pi_\theta \,\|\, \pi_\alpha\big) - \log Z_\alpha(z)\Big],$$

where $Z_\alpha(z)$ is the normalizing constant. Since $\beta > 0$ only rescales the objective and $\log Z_\alpha(z)$ does not depend on θ, maximizing $J_{\text{KL-RL}}$ is equivalent to minimizing the reverse KL divergence to π_α. Neither β nor the intractable Z_α enters the optimization.

RLOO Training Objective. Factorizing both log-probabilities with the chain rule reveals a token-level reward structure, which we exploit through several variance-reduction techniques to derive a REINFORCE reward-to-go (G_t) estimator. Incorporating a leave-one-out baseline, b_t then yields our final gradient estimator;

$$\nabla_\theta \mathcal{L} = \mathbb{E}\left[\sum_t \nabla_\theta \log \pi_\theta(x_t \mid z, x_{<t}) \cdot (G_t - b_t)\right], \tag{11}$$

with full details and derivation provided in Appendix A.

Efficient fine-tuning by warm start. The power conditional (9) factorizes as $\pi_\alpha(x_t \mid z, x_{<t}) \propto \pi_0(x_t \mid z, x_{<t})^\alpha\, \zeta_t(x_t)$: a locally tempered term, times the lookahead twist ζ_t. We therefore sample and score the fine-tuned model at temperature $1/\alpha$ throughout, giving

$$\pi_\theta(x_t \mid z, x_{<t}) \propto \exp\big(\alpha\,[\ell_0(x_t) + \Delta\ell_{\text{LoRA}}(x_t)]\big), \tag{12}$$

where ℓ_0 are the base model's logits and $\Delta\ell_{\text{LoRA}}$ is the adapter's contribution. At initialization ($\Delta\ell_{\text{LoRA}} = 0$), $\pi_\theta(x_t \mid z, x_{<t}) \propto \pi_0(x_t \mid z, x_{<t})^\alpha$, and matching it to π_α shows that the adapter need only learn

$$\Delta\ell_{\text{LoRA}}(v) = \tfrac{1}{\alpha} \log \zeta_t(v) + \underbrace{c(z, x_{<t})}_{\text{constant}}, \tag{13}$$

i.e the hard to approximate lookahead *twist* term. Appendix B shows that, under this parametrization, the cost-to-go reduces to the cumulative Rényi entropy of future continuations at initialization. Appendix C demonstrates that our warm start requires substantially less training, as we begin closer to the target distribution.

Benefits of POWER-RLOO

1. **Measurable fidelity.** Because π_θ is an explicit model, we can evaluate its likelihoods and track the loss $\mathcal{L}(\theta) = D_{\text{KL}}(\pi_\theta \,\|\, \pi_\alpha) - \log Z_\alpha$ throughout training. This directly measures how closely we approach the target, rather than inferring it from an approximate sampler.
2. **Clean sensitivity analysis.** This lets us isolate the effect of α, both on the learned distribution itself and on downstream sample quality.
3. **Simplicity.** Training is straightforward to implement and requires none of the engineering machinery (block proposals, resampling, cache management) required by MCMC- and SMC-based approaches.

5 EXPERIMENTS

We can now, for the first time, concretely assess the value of targeting the power distribution. We deploy POWER-RLOO, POWER-SMC, and additional heuristics across a range of models and two benchmarks. Full code will be released upon acceptance.

5.1 SETUP

Models and Datasets. We evaluate on Qwen2.5-7B (Base and Instruct) and Phi-3.5-mini-Instruct. We use MATH500 (Lightman et al., 2023), a set of 500 competition-level mathematics problems spanning algebra, geometry, number theory, and combinatorics, to assess multi-step mathematical reasoning. We additionally evaluate on the GPQA Diamond dataset (Rein et al., 2023), a set of graduate-level, multiple-choice questions in biology, physics, and chemistry written by domain experts to be resistant to simple web lookup.

Generation Config and Hyperparameters. To target the power distribution as faithfully as possible, we disable top-p and top-k sampling and remove any repetition penalties, ensuring that the distribution we sample from (and compare against) remains unaltered. We follow existing work, and focus on targeting $\alpha = 4$. For fine-tuning, we use LoRA (with rank 32) and train in stages across $\alpha \in \{1.5, 2, 3, 4\}$, training for 200 steps at each value of α with a learning rate of 1e-4. Due to compute restraints we limit generation length to 1564. All training was done on a single Nvidia H200 GPU. Comprehensive hyperparameters are detailed in Appendix D.

Baselines. As motivated in Section 3, we implement a simple resampling scheme in which, after every block of 64 tokens, we select the top-k samples by log-probability and duplicate them; we refer to this as Best-of-N w/ Resampling. We report values with $k = 4$ and $N = 16$. We additionally compare against standard Best-of-N (Huang et al., 2024), where we generate N low temperature samples ($\tau = 0.25$) and select the highest-scoring sequence according to either total or mean log-probability:

$$\hat{x}_{\text{sum}} = \arg\max_x \sum_t \log p_\theta(x_t \mid z, x_{<t}),$$

$$\hat{x}_{\text{mean}} = \arg\max_x \frac{1}{|x|} \sum_t \log p_\theta(x_t \mid z, x_{<t}).$$

Model	Method	MATH500 (%)	GPQA (%)
Qwen2.5-7B-Base	Baseline decoding	39.0	28.6
	Low-temperature decoding ($\tau = 1/\alpha$)	63.1	31.4
	Power-SMC	**70.6**	31.9
	Best-of-N-mean	67.2	**32.2**
	Best-of-N-sum	62.6	32.2
	Best-of-N w/ Resampling	**70.0**	29.7
	Power-RLOO (ours)	N/A	N/A
Qwen2.5-7B-Instruct	Baseline decoding	73.1	35.9
	Low-temperature decoding ($\tau = 1/\alpha$)	75.4	35.1
	Power-SMC	76.9	35.8
	Best-of-N-mean	77.0	36.5
	Best-of-N-sum	75.9	36.5
	Best-of-N w/ Resampling	76.9	32.6
	Power-RLOO (ours)	**78.4**	**41.0**
Phi-3.5-mini-instruct	Baseline decoding	38.9	29.1
	Low-temperature decoding ($\tau = 1/\alpha$)	41.7	**29.4**
	Power-SMC	**46.6**	28.0
	Best-of-N-mean	43.8	28.7
	Best-of-N-sum	43.8	28.6
	Best-of-N w/ Resampling	45.6	26.5
	Power-RLOO (ours)	43.6	29.2

Table 1: Pass@1 accuracy on MATH500 and GPQA at $\alpha = 4$. POWER-RLOO and Power-SMC yield only modest, model-dependent gains over simpler mode-seeking baselines such as Best-of-N. N/A denotes a lack of convergence due to an undesirable length degeneracy pathology arising from more precisely targeting the power distribution (see Section 5.2)

5.2 RESULTS AND ANALYSIS

We report accuracy across all three models and both benchmarks in Table 1, observing the following notable patterns in line with the intuition developed by Figure 1.

Sharpening works best on top of a strong prior. We find that a faithful representation of the power distribution is only beneficial when the underlying base model is already reasonably well-aligned with the task at hand. This is most evident on Qwen2.5-7B-Instruct, where the more accurate sharpening achieved by POWER-RLOO leads to the strongest results of any method on both benchmarks (78.4% on MATH500 and 41.0% on GPQA). By contrast, on the weaker Phi-3.5-mini-Instruct model, POWER-RLOO trails Power-SMC on MATH500 (43.6% vs. 46.6%), suggesting that sharpening only helps if there is a meaningful signal in the base distribution to sharpen toward.

Standard Best-of-N frequently matches or exceeds Power-SMC. For GPQA, Best-of-N-mean outperforms Power-SMC for every model (e.g. Qwen-Base: 32.2% vs. 31.9%; Qwen-Instruct: 36.5% vs. 35.8%; Phi: 28.7% vs. 28.0%), and even the more accurate sharpening achieved by our POWER-RLOO. On MATH500, Best-of-N-mean matches Power-SMC for Qwen-Instruct (77.0% vs. 76.9%). These results reinforce our core message: much of the benefit attributed to SMC-based sharpening is achievable through simple, training-free selection.

Benefits over low temperature sampling are not guaranteed. For MATH500, the additional gain from low temperature sampling to Power-RLOO or SMC is sometimes sizeable (e.g. Qwen-Base: $63.1\% \rightarrow 70.6\%$ with Power-SMC), however, on GPQA the gap between low-temperature decoding and the more sophisticated methods closes. Therefore, the gains of sharpening vary significantly by task structure.

Faithfully targeting π_α leads to length degeneracy. Shorter trajectories carry higher per-sequence probability and are consequently favored by the power distribution. On Qwen-Base, POWER-RLOO

collapsed toward degenerate, overly short completions (denoted N/A in Table 1). Notably, SMC approximations and Best-of-N avoid this problem, precisely because they fail to converge to π_α. This reveals an important trade-off: faithful approximation of π_α exposes pathological behavior that inference-time samplers sidestep through convergence failure. Maintaining reasonable output length under sharpening requires a strong prior over length itself – one such as that instilled via chat-template tuning in instruct models.

Weak sharpening uncovers latent knowledge. Sharpening naturally reduces the typical set of sampled trajectories, and a faithful approximation of the power distribution should reflect this. Indeed, as shown in Figure 4, POWER-RLOO's pass@k decreases with increasing α; however, mild sharpening actually improves this metric at large k, suggesting it surfaces latent knowledge already present in the model but rarely sampled at $\alpha = 1$. This reveals a practically valuable property of sharpening: while aggressive mode-seeking optimizes raw accuracy, mild sharpening exposes existing reasoning capacity—arguably a more useful outcome than further narrowing the distribution. Power-SMC's poor pass@k stems from particle degeneracy; we report pass@k over re-weighted particles from a single run rather than k independent populations, which would be prohibitively expensive.

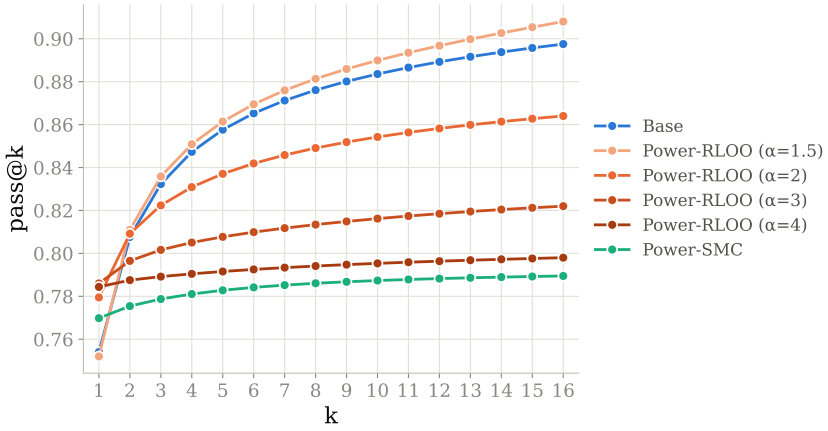


Figure 4: Pass@k accuracy for POWER-RLOO and Power-SMC ($\alpha = 4$) on Qwen2.5-7B-Instruct (MATH500). As expected, pass@k decreases with α; however, mild sharpening ($\alpha = 1.5$) actually improves pass@k, suggesting it elicits reasoning paths that exist in the base model but are rarely sampled at $\alpha = 1$

6 CONCLUSION

We have critically examined recently proposed MCMC and SMC schemes for targeting an LLM's power distribution, showing that they fail to converge and instead behave as likelihood-guided search. We then introduce POWER-RLOO, a principled fine-tuning algorithm that directly targets π_α and allows, for the first time, an precise investigation into the common claim that sharpening drives reasoning improvement.

We found that faithful approximation of π_α yields only modest, model-dependent gains, and sometimes undesirable phenomenon, like length degeneracy. Best-of-N matches or exceeds sharpening without targeting π_α, demonstrating that improvements stem from selection, not approximation quality. The field should reconsider power sampling as a general path to reasoning improvement.

AI USE STATEMENT

In this work, we did not use generative AI tools for any of the tasks requiring disclosure (generating synthetic datasets; developing theoretical models or conceptual frameworks; formulating mathematical claims; providing critical ingredients for proving mathematical claims; assisting in the writing of proofs; proposing or refining hypotheses; designing or providing feedback on research methodology or experiments; implementing methods; assisting with translation; cleaning and reformatting datasets; supporting qualitative and thematic data analysis; or interpreting results). Additionally, we used generative AI tools for creating/modifying scientific figures (plotting help) and editing the research paper to improve readability (proofreading). We have reviewed all AI-assisted work. All AI-suggested plots were checked against the underlying data by the authors for accuracy before inclusion, and all AI-suggested text edits were manually reviewed against the original meaning to ensure no content or claims were altered. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work focuses on improving the data efficiency of large language model training, and does not involve human subjects, new data collection, or the release of a new dataset. We do not anticipate significant risks of harm arising directly from our method. To the extent that our approach reduces the amount of data and compute required to train performant models, it may have a modest positive environmental impact by lowering the computational cost of training. As with any advance that improves the capability or accessibility of LLMs, we acknowledge the general dual-use consideration that more efficient training could also lower the barrier to developing models for harmful purposes; we believe this risk is not unique to our method and is outweighed by the broader benefits of more efficient, less resource-intensive model development. We are not aware of any conflicts of interest or sponsorship arrangements relevant to this work.

REPRODUCIBILITY STATEMENT

Experimental settings are provided in Section 5, and full code will be published upon acceptance.

REFERENCES

Seyedarmin Azizi, Erfan Baghaei Potraghloo, Minoo Ahmadi, Souvik Kundu, and Massoud Pedram. Power-smc: Low-latency sequence-level power sampling for training-free llm reasoning, 2026. URL https://arxiv.org/abs/2602.10273.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

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. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Andre He, Daniel Fried, and Sean Welleck. Rewarding the unlikely: Lifting grpo beyond distribution sharpening, 2025. URL https://arxiv.org/abs/2506.02355.

Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. Large language models for software engineering: A systematic literature review, 2024. URL https://arxiv.org/abs/2308.10620.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model, 2025. URL https://arxiv.org/abs/2503.24290.

Audrey Huang, Adam Block, Dylan J. Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, Jordan T. Ash, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism, 2024. URL https://arxiv.org/abs/2412.01951.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve, 2022. URL https://arxiv.org/abs/2210.11610.

Xiaotong Ji, Rasul Tutunov, Matthieu Zimmer, and Haitham Bou Ammar. Scalable power sampling: Unlocking efficient, training-free reasoning for llms via distribution sharpening, 2026. URL https://arxiv.org/abs/2601.21590.

Aayush Karan and Yilun Du. Reasoning with sampling: Your base model is smarter than you think, 2025. URL https://arxiv.org/abs/2510.14901.

Tomasz Korbak, Ethan Perez, and Christopher L Buckley. Rl with kl penalties is better viewed as bayesian inference, 2022. URL https://arxiv.org/abs/2205.11275.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, December 2022. ISSN 1095-9203. doi: 10.1126/science.abq1158. URL http://dx.doi.org/10.1126/science.abq1158.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof qa benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. Spurious rewards: Rethinking training signals in rlvr, 2026. URL https://arxiv.org/abs/2506.10947.

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, 2024. URL https://arxiv.org/abs/2402.03300.

KE WANG, ZEHAO YU, Luwei Wang, and Yongchao Huang. Cache coherent resampling for efficient test time scaling in LLM reasoning via adaptive sequential monte carlo. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=JN6wxUGmW8.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Reinforcement learning for reasoning in large language models with one training example, 2025. URL https://arxiv.org/abs/2504.20571.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2023. URL https://arxiv.org/abs/2212.10560.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models, 2025. URL https://arxiv.org/abs/2401.10020.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025. URL https://arxiv.org/abs/2504.13837.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards, 2026. URL https://arxiv.org/abs/2505.19590.

Matthieu Zimmer, Xiaotong Ji, Rasul Tutunov, Anthony Bordg, Jun Wang, and Haitham Bou Ammar. Bourbaki: Self-generated and goal-conditioned mdps for theorem proving, 2025. URL https://arxiv.org/abs/2507.02726.

A Deriving the Training Objective

We begin from the Kl informed loss

$$J_{\text{KL-RL}}(\theta) = \beta\, \mathbb{E}_{x\sim\pi_\theta(\cdot|z)}\big[\alpha \log \pi_0(x \mid z) - \log \pi_\theta(x \mid z)\big] = -\beta\Big[D_{\text{KL}}\big(\pi_\theta \,\|\, \pi_\alpha\big) - \log Z_\alpha(z)\Big]. \tag{14}$$

Factorizing both log-probabilities with the chain rule, we obtain the training loss

$$\mathcal{L}(\theta) \;=\; \mathbb{E}_{x\sim\pi_\theta(\cdot|z)}\left[\sum_{t=1}^{T} f_t\right], \qquad f_t \;\coloneqq\; \log \pi_\theta(x_t \mid z, x_{<t}) \;-\; \alpha \log \pi_0(x_t \mid z, x_{<t}), \tag{15}$$

which we minimize. Because $\mathcal{L}(\theta) = D_{\text{KL}}(\pi_\theta \,\|\, \pi_\alpha) - \log Z_\alpha(z)$, its unique minimizer is $\pi_\theta = \pi_\alpha$.

We now derive POWER-RLOO's training objective using the following identity.

Lemma 1 (Zero-mean conditional score)**.** *Let Y be any random variable that is conditionally independent of x_t given the prefix $x_{<t}$. Then*

$$\mathbb{E}_{x\sim\pi_\theta(\cdot|z)}[\, Y\, \nabla_\theta \log \pi_\theta(x_t \mid z, x_{<t})\,] \;=\; 0. \tag{16}$$

Proof. Condition on the prefix $y_{<t}$ and apply the tower rule:

$$\mathbb{E}[Y\, \nabla_\theta \log \pi_\theta(x_t \mid z, x_{<t})] = \mathbb{E}_{x_{<t}}\big[\, \mathbb{E}[Y \mid x_{<t}] \cdot \mathbb{E}_{x_t|x_{<t}}[\nabla_\theta \log \pi_\theta(x_t \mid z, x_{<t})]\big]\,,$$

where the factorization of the inner expectation follows from the conditional independence of Y and y_t given $y_{<t}$. The inner score term has zero mean:

$$\begin{aligned}\sum_{v\in\mathcal{V}} \pi_\theta(v \mid z, x_{<t})\, \nabla_\theta \log \pi_\theta(v \mid z, x_{<t}) &= \sum_{v\in\mathcal{V}} \nabla_\theta \pi_\theta(v \mid z, x_{<t}) \\ &= \nabla_\theta \sum_{v\in\mathcal{V}} \pi_\theta(v \mid z, x_{<t}) = \nabla_\theta 1 = 0,\end{aligned}$$

where $\mathcal{V}$ denotes the vocabulary, so that summing over v ranges over all possible next tokens. □

Differentiating Equation 15 via the product rule gives

$$\nabla_\theta \mathcal{L} = \underbrace{\mathbb{E}\Big[\Big(\textstyle\sum_s f_s\Big)\, \nabla_\theta \log \pi_\theta(x \mid z)\Big]}_{\text{distributional}} + \underbrace{\mathbb{E}[\textstyle\sum_t \nabla_\theta f_t]}_{\text{pathwise}}. \tag{17}$$

Step 1: the pathwise term vanishes. Since $\alpha \log \pi_0(x_t \mid z, x_{<t})$ does not depend on θ, we have $\nabla_\theta f_t \;=\; \nabla_\theta \log \pi_\theta(x_t \;\mid\; z, x_{<t})$, so Lemma 1 (with $Y \;=\; 1$) annihilates each summand. This cancellation holds *in expectation*, not per sample: dropping the pathwise term discards a mean-zero source of gradient noise, rather than a source of bias.

Step 2: the past drops out (reward-to-go). Expanding $\nabla_\theta \log \pi_\theta(x \mid z) = \sum_t \nabla_\theta \log \pi_\theta(x_t \mid z, x_{<t})$ in the distributional term leaves a double sum $\sum_t \sum_s \mathbb{E}\big[f_s\, \nabla_\theta \log \pi_\theta(x_t \mid z, x_{<t})\big]$. For any pair with $s < t$, the quantity f_s is a function only of $x_{\le s} \subseteq x_{<t}$, and is therefore conditionally independent of x_t given $x_{<t}$; Lemma 1 (with $Y = f_s$) kills each such term. Only the terms with $s \ge t$ survive, leaving

$$\nabla_\theta \mathcal{L} = \mathbb{E}\left[\sum_t \nabla_\theta \log \pi_\theta(x_t \mid z, x_{<t}) \cdot G_t\right], \qquad G_t \;\coloneqq\; \sum_{s\ge t} f_s. \tag{18}$$

Step 3: subtract a baseline. Lemma 1 with $Y = b_t$, for any b_t conditionally independent of x_t given $y_{<t}$, gives the final estimator:

$$\boxed{\nabla_\theta \mathcal{L} = \mathbb{E}\left[\sum_t \nabla_\theta \log \pi_\theta(y_t \mid z, x_{<t}) \cdot \big(G_t - b_t\big)\right], \qquad A_t \coloneqq G_t - b_t.} \tag{19}$$

We use the leave-one-out batch mean at each position,

$$b_{i,t} = \frac{1}{B-1} \sum_{j \neq i} G_{j,t}, \tag{20}$$

In practice, one does not differentiate Equation 19 by hand. Instead, we form the surrogate scalar

$$\widehat{\mathcal{L}}(\theta) = \frac{1}{B} \sum_{i=1}^{B} \sum_{t=1}^{T} \mathrm{sg}\big(\hat{A}_{i,t}\big)\ \log \pi_\theta(x_{i,t} \mid z_i, x_{i,<t}), \tag{21}$$

where $\mathrm{sg}(\cdot)$ denotes the stop-gradient (`detach`) operator, and call `backward` on it directly.

B Connection to Rényi entropy

Our low-temperature parameterization also gives the training signal a clean interpretation. Write $Z_t = \sum_v \pi_0(v \mid z, x_{<t})^\alpha$ and let $H_\alpha(p) = \frac{1}{1-\alpha} \log \sum_v p(v)^\alpha$ denote the Rényi entropy of order α. At initialization, $\pi_\theta = \pi_0^\alpha / Z_t$, and the per-token cost in Equation (15) reduces to

$$f_t = \log \pi_\theta(x_t \mid z, x_{<t}) - \alpha \log \pi_0(x_t \mid z, x_{<t}) = -\log Z_t = (\alpha - 1)\, H_\alpha\big(\pi_0(\cdot \mid z, x_{<t})\big). \tag{22}$$

Notably, this cost does not depend on the sampled token x_t, only on the entropy of the distribution it was drawn from. The cost-to-go is therefore the cumulative Rényi entropy along the rest of the rollout,

$$G_t = (\alpha - 1) \sum_{s \geq t} H_\alpha\big(\pi_0(\cdot \mid z, x_{<s})\big). \tag{23}$$

Since the $s = t$ term is fixed by the prefix, the influence assigned to x_t comes entirely from the entropy of its continuations. This observation mirrors the role of the twist in Equation (13). Because we minimize $\mathcal{L}$, tokens whose continuations are less uncertain than the leave-one-out baseline have their logits raised, and those leading into more uncertain continuations have them lowered.

C Low Temperature Sampling in Power-RLOO

In this section we explore the utility of always using low-temperature sampling when training with Power-RLOO. Appendix B gives a clean mathematical interpretation of the rewards under low-temperature sampling; here we show that low-temperature sampling also yields smoother and faster convergence to the target.

Because Power-RLOO maintains a valid distribution throughout training, we can directly observe training dynamics via estimates of the KL divergence. As shown in Figure 5 (global step 0), the low-temperature sampling distribution starts closer to the power distribution in terms of KL. This is expected: Equation (9) shows that the two are inextricably linked via the twist term learned by the LoRA adapter, Equation (13). We also observe that the warm start transfers across jumps in α, avoiding the sharp KL jump encountered otherwise. This may be interpreted as evidence that the twist learned for a smaller α is a good starting approximation for the twist required at a larger α.

D Implementation Details

Model and PEFT. We conduct experiments with three base models: `Qwen/Qwen2.5-7B-Instruct`, `Qwen/Qwen2.5-7B` (base), and `Phi-3.5-mini-instruct`, all loaded in bf16 with `sdpa` attention. LoRA is applied with rank

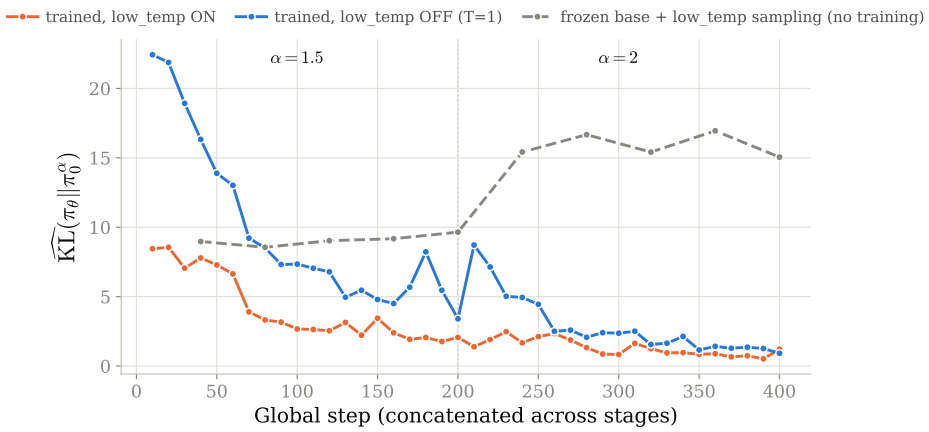


Figure 5: Progression of KL divergence to the target distribution over the course of training (lower is better). By maintaining a valid probability distribution, we are able to compute estimates of KL to track training stability. We observe faster and more stable convergence when using low-temperature sampling.

16, $\alpha = 32$, dropout 0, no bias, targeting all attention and MLP projections (`q/k/v/o_proj`, `gate/up/down_proj`). The reference policy π_0 is the same model with the adapter disabled, avoiding a second resident copy of the base model. The LM head is cast to fp32 before PEFT wrapping via a custom pre-PEFT hook, since TRL's built-in cast breaks on PEFT-wrapped models.

Batching and sampling. Each optimizer step uses 32 completions per question accumulated over 10 questions, for an effective batch of 320 rollouts per step; one micro-batch corresponds to one question's group, shuffled across the dataset. Prompts are capped at 768 tokens and completions at 1564 tokens. Generation uses a colocated vLLM engine (GPU memory utilization 0.15); truncated importance sampling correction is disabled, as π_0 rescoring under Hugging Face is treated as ground truth and the vLLM/HF discrepancy is attributable to bf16 kernel noise (verified at the start of training to be within 0.15 nats/token). No-grad old-policy and teacher log-probability passes use Apple's Cut Cross Entropy to avoid materializing the full (B, T, V) logits tensor; the gradient-carrying forward pass inside the TRL loss remains on the standard path.

Optimization. We use a learning rate of $1\mathrm{e}{-4}$ on LoRA parameters only, with no weight decay, Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$), and gradient-norm clipping at 1.0. The learning-rate schedule is per-stage rather than global: cosine decay to zero with a single restart per stage, 3% warmup of that stage's steps, and a minimum LR ratio of 0; optimizer state is reset at each stage boundary along with a fresh warmup. We use `num_iterations=1`, i.e. a purely on-policy REINFORCE estimator with no PPO-style replay or clipping.

Other. A single adapter is trained across the entire α schedule, without rebuilding the trainer or vLLM engine between stages. All runs use seed 0.