Introduction to Generative AI Models
Final Project Report

Selim-Antoine Lali, Gabriel Bus, Théo Cadène
research paper analysis: github.com/SelimLali/LLM-reasoning

Reasoning with Sampling: Your Base Model is Smarter Than You Think
Aayush Karan and Yilun Du (2025), https://arxiv.org/pdf/2510.14901

January 19, 2026

Abstract

Frontier "reasoning" language models are often produced by post-training base LLMs with reinforcement learning (RL), particularly RL with verifiable rewards. A central open question is whether RL creates fundamentally new reasoning behaviors, or whether it mainly *sharpens* the base model distribution toward already-existing high-likelihood reasoning traces. The paper *Reasoning with Sampling: Your Base Model is Smarter Than You Think* proposes a training-free inference-time alternative: sample from a *power distribution* p^α derived from the base model p, using an approximate Metropolis–Hastings (MH) procedure adapted to autoregressive generation. Empirically, this "power sampling" method nearly matches—and sometimes outperforms—RL post-training (GRPO) on diverse benchmarks (MATH500, HumanEval, GPQA, AlpacaEval 2.0), while avoiding the diversity collapse often observed in RL-posttrained models. This report summarizes the motivations, method, theoretical justification, and experimental findings of the paper, and discusses its implications for inference-time scaling in generative models.

Contents

1 Introduction

Large language models (LLMs) are generative models over token sequences. Recent "reasoning models" demonstrate strong performance in mathematics, coding, and scientific QA, often attributed to post-training with reinforcement learning (RL), especially RL with verifiable rewards. Despite strong empirical gains, an active debate asks whether RL produces *new* reasoning capabilities or mostly performs *distribution sharpening*, i.e., reallocating probability mass toward already-present high-likelihood reasoning traces.

The analyzed paper takes a complementary view: *can we elicit comparable reasoning from the base model at inference time, purely via sampling, without additional training or verifiers ?* The authors propose:

- a **target distribution** for sampling: the *power distribution* p^α (for $\alpha > 1$),
- an **approximate sampling algorithm** based on MH resampling with an autoregressive block schedule,
- an **empirical evaluation** showing near-RL single-shot reasoning improvements while preserving multi-sample diversity (pass@k).

Contributions

1. Formalize **power distributions** as a principled sharpening target for base-model reasoning.
2. Develop a **training-free MH-based sampler** for autoregressive LLMs that approximates sampling from p^α.
3. Demonstrate strong performance across models and benchmarks, including **out-of-domain** tasks for RL post-training.

Organization of my report: In Section 3, I introduce the notations. Section 4 presents power distributions and the sampler, with pseudocode and an illustrative example. Section 4.3 explains correctness intuitions and key theoretical claims (including why temperature sampling is not equivalent). Finally, Section 5 summarizes the paper's evaluation protocol and results, including diversity analysis. We conclude with discussion and limitations.

2 Related Work

RL with human feedback (RLHF) and RL with verifiable rewards (RLVR) are dominant post-training paradigms for improving alignment and reasoning. GRPO is highlighted in the paper as a standard algorithm used to post-train reasoning models.

A second relevant line of work adapts classical sampling techniques (MCMC, SMC, tempering/annealing) to generative models. In diffusion models, annealed/tempered distributions are used to improve sample quality or explore multi-modal targets; the paper draws an analogy and uses similar "distribution sharpening" ideas for LLM sampling.

This report focuses on the paper's core claim: *a surprisingly large fraction of apparent RL reasoning gains can be matched by inference-time sampling from the base model.*

3 Preliminaries and Notation

Let $\mathcal{X}$ be a finite vocabulary and $\mathcal{X}^T$ be the set of token sequences $x_{0:T} = (x_0, \dots, x_T)$. An autoregressive LLM defines a distribution p over sequences via:

$$p(x_{0:T}) = \prod_{t=0}^{T} p(x_t \mid x_{<t}). \tag{1}$$

Standard generation samples sequentially from the conditional next-token distributions $p(x_t \mid x_{<t})$.

Goal. Given only a base model p, design an inference-time procedure that returns a *single* final response that is more likely to be correct on reasoning tasks, without training, curated data, or external reward/verifier.

4 Problem Definition, Methodology, and Theoretical Analysis

4.1 Task Definition

The paper's tasks are *single-shot* reasoning benchmarks: for each prompt, the model outputs one response; accuracy is computed by a task-specific verifier (e.g., exact math answer match, unit tests for code, multiple-choice correctness). Additionally, an *unverifiable* setting (AlpacaEval 2.0) uses an LLM judge to score helpfulness.

The methodological question is:

Can we improve single-shot reasoning by reshaping the sampling distribution implied by p, using only base-model likelihoods and additional inference-time compute?

4.2 Presentation of the Method

4.2.1 Power Distributions as Explicit Distribution Sharpening

A natural family of "sharpened" distributions derived from a base distribution p is the *power distribution*:

$$\pi(x) \propto p(x)^{\alpha}, \qquad \alpha \geq 1. \tag{2}$$

For $\alpha > 1$, relative probability mass increases on higher-likelihood sequences.

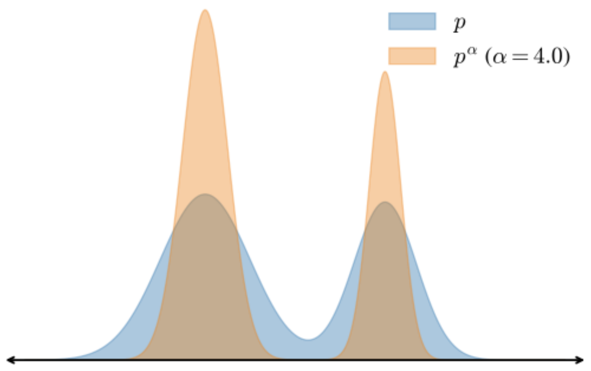

Figure 1: Visualization of distribution sharpening: a base density p and its power/tempered version p^α for $\alpha > 1$.

Why not just use lower temperature? A common heuristic is *low-temperature sampling*, which modifies conditionals:

$$p_{\text{temp}}(x_t \mid x_{<t}) = \frac{p(x_t \mid x_{<t})^\alpha}{\sum_{x' \in \mathcal{X}} p(x' \mid x_{<t})^\alpha}, \qquad \tau = \frac{1}{\alpha}. \tag{3}$$

However, the key theoretical point is that (3) *does not* sample from the joint power distribution (2). This difference matters for reasoning, because the true power distribution implicitly accounts for the quality of *future completions*.

4.2.2 Key Proposition: Temperature Sampling $\neq$ Power Distribution Sampling

Proposition 1 (Low-temperature sampling does not sample from p^α)**.** *For an autoregressive model, sampling tokens sequentially from the tempered conditionals in* (3) *does not, in general, produce samples from the joint power distribution* $\pi(x) \propto p(x)^\alpha$.

Intuition. Under p^α, the conditional weight of a token x_t depends on a *sum of exponentiated probabilities of all future completions* consistent with choosing x_t. Low-temperature sampling instead takes an *exponent of a sum* (a greedy local reweighting), which does not match the required marginalization structure. This distinction can prefer tokens with many mediocre continuations over tokens with fewer high-quality continuations, which is undesirable in "critical window" reasoning scenarios.

4.2.3 A Concrete Example (Critical Window Intuition)

Example 1 (Two-token sequences where p^α differs from low-temperature)**.** *Consider* $\mathcal{X} = \{a, b\}$ *and sequences of length 2:* aa, ab, ba, bb. *Let*

$$p(aa) = 0.00, \quad p(ab) = 0.40, \quad p(ba) = 0.25, \quad p(bb) = 0.25.$$

For $\alpha = 2$*, the true power distribution prefers* $x_0 = a$ *(because it upweights the strong future path* ab*), while low-temperature prefers* $x_0 = b$ *(because* b *has two average-quality futures).*

4.2.4 Sampling from an Unnormalized Target: Metropolis–Hastings

Direct sampling from $\pi(x) \propto p(x)^\alpha$ is intractable because normalization requires summing over $\mathcal{X}^T$. MH constructs a Markov chain with stationary distribution π using a proposal $q(x' \mid x)$ and acceptance probability:

$$A(x', x) = \min\left(1, \frac{\pi(x')\, q(x \mid x')}{\pi(x)\, q(x' \mid x)}\right). \tag{4}$$

Algorithm 1 Power Sampling for Autoregressive Models (paper's Algorithm 1)

```
Require: Base LLM $p$, proposal LLM $p_{\text{prop}}$, power $\alpha$, max length $T$
Require: Block size $B$, MH steps $N_{\text{MCMC}}$
 1: Define unnormalized targets $\pi_k(x_{0:kB}) \propto p(x_{0:kB})^\alpha$
 2: for $k = 0$ to $\lceil T/B \rceil - 1$ do
 3:     Initialize by extending current prefix $x_{0:kB}$ with proposal sampling:
          for $t = kB + 1$ to $(k + 1)B$: sample $x_t^{(0)} \sim p_{\text{prop}}(\cdot \mid x_{<t})$
 4:     Set current state $x \leftarrow x^{(0)}$
 5:     for $n = 1$ to $N_{\text{MCMC}}$ do
 6:         Sample an index $m$ uniformly from $\{1, \dots, (k + 1)B\}$
 7:         Construct proposal $x'$ by keeping prefix $x'_{0:m-1} = x_{0:m-1}$ and resampling suffix:
              for $t = m$ to $(k + 1)B$: sample $x'_t \sim p_{\text{prop}}(\cdot \mid x'_{<t})$
 8:         Compute acceptance:
              $$A(x', x) \leftarrow \min\left(1, \frac{\pi_{k+1}(x')}{\pi_{k+1}(x)} \cdot \frac{p_{\text{prop}}(x \mid x')}{p_{\text{prop}}(x' \mid x)}\right)$$
 9:         Draw $u \sim \text{Uniform}(0, 1)$
10:         if $u \le A(x', x)$ then
11:             accept: $x \leftarrow x'$
12:         end if
13:     end for
14:     Fix the new prefix: $x_{0:(k+1)B} \leftarrow x$
15: end for
16: return $x_{0:T}$
```

If q is irreducible and aperiodic, the chain converges to π.

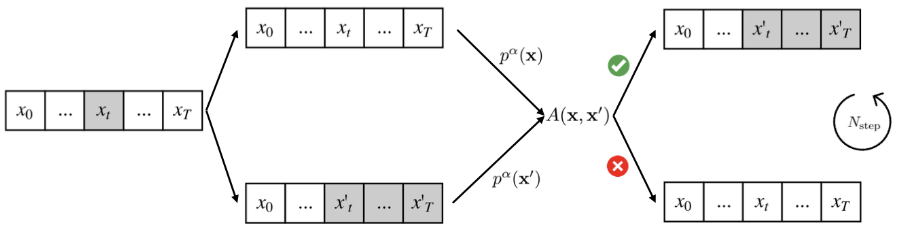

Figure 2: MH with random resampling: choose an index, resample a suffix with a proposal model, then accept/reject using the power-likelihood ratio.

4.2.5 Autoregressive Blockwise MH: Power Sampling Algorithm

A direct MH chain over full-length responses can mix poorly. The paper proposes a blockwise scheme: sample progressively longer prefixes with intermediate targets

$$\pi_k(x_{0:kB}) \propto p(x_{0:kB})^\alpha, \tag{5}$$

where B is a block size. Each stage runs MH resampling within the prefix length $(k + 1)B$.

Inference-time scaling cost. The paper estimates the expected number of tokens generated by resampling as approximately

$$\mathbb{E}[\#\text{generated tokens}] \approx \frac{N_{\text{MCMC}} T^2}{4B}. \tag{6}$$

Thus, N_{MCMC} provides a natural inference-time compute knob.

4.2.6 Trace-through Example - How the Algorithm Behaves

Let's give a short, intuitive trace for a prompt (e.g., a math problem):

1. **Stage 1**: sample an initial prefix of length B using p_{prop} (e.g., low temperature).
2. **MH steps**: pick a random position m within the prefix, resample the suffix, and accept if the new completion increases the powered likelihood ratio.
3. **Later stages**: extend to $2B$, then $3B$, etc., each time using the previously accepted prefix as warm start.

It is like a "local editing" of a candidate solution, with accept/reject driven by base-model likelihood under p^α.

4.3 Theoretical Justification and Guarantees

4.3.1 Correctness of MH Targeting p^α

MH is a standard method to sample from an unnormalized target distribution. If we define $\pi(x) \propto p(x)^\alpha$ and use a proposal kernel $q(x' \mid x)$ satisfying mild conditions (irreducibility, aperiodicity), then the Markov chain's stationary distribution is π. The acceptance ratio (4) ensures detailed balance with respect to π.

4.3.2 Why Power Distributions Help Reasoning

The paper's key intuition is that reasoning failures often involve *pivotal tokens* (critical windows): early choices that keep average likelihood high but lead to poor individual completions. Power distributions can upweight tokens that yield fewer but higher-likelihood futures, effectively introducing a planning-like bias at token choice time.

Observation 1 (Qualitative effect of p^α vs temperature sampling)**.** *Power distribution sampling upweights tokens associated with few high-probability future paths, whereas low-temperature sampling can prefer tokens with many average-probability completions.*

4.3.3 Proof Sketch for Proposition 1

We include the proof in a report-friendly form.

Proof sketch. For the joint power distribution $\pi(x) \propto p(x)^\alpha$, the correct conditional for token x_t is

$$p_{\text{pow}}(x_t \mid x_{<t}) = \frac{\sum_{x_{>t}} p(x_{0:T})^\alpha}{\sum_{x_{\geq t}} p(x_{0:T})^\alpha},$$

which is a *sum of exponentiated path probabilities.* In contrast, low-temperature sampling uses

$$p_{\text{temp}}(x_t \mid x_{<t}) \propto \left(\sum_{x_{>t}} p(x_{0:T})\right)^\alpha,$$

an *exponent of a sum* induced by locally tempering next-token probabilities. These generally differ unless the future has degenerate structure; hence the induced joint distributions differ. □

Practical implication. Matching RL-like distribution sharpening requires considering future paths more globally than a local temperature change does; MH resampling is one way to approximate that effect without training.

5 Experimental Evaluation

5.1 Methodology

The paper evaluates single-shot performance across:

- **MATH500**: subset of MATH test problems (competition math).
- **HumanEval**: 164 coding tasks evaluated by unit tests.
- **GPQA (Diamond)**: difficult multiple-choice science questions.
- **AlpacaEval 2.0**: general helpfulness judged by an automated LLM judge.

Models and baselines. Three base LLMs are used: Qwen2.5-Math-7B, Qwen2.5-7B, and Phi-3.5-mini-instruct. The RL baseline is GRPO post-training (trained on MATH training split). Sampling baselines include standard base sampling and low-temperature sampling.

Power sampling hyperparameters The paper uses max length $T_{\max} = 3072$, block size $B = 192$, and typically $\alpha = 4.0$ with a proposal distribution chosen as the base model with temperature $1/\alpha$. For AlpacaEval 2.0, a higher-temperature proposal (e.g. $\tau = 0.5$) is reported to help.

5.2 Quantitative Results

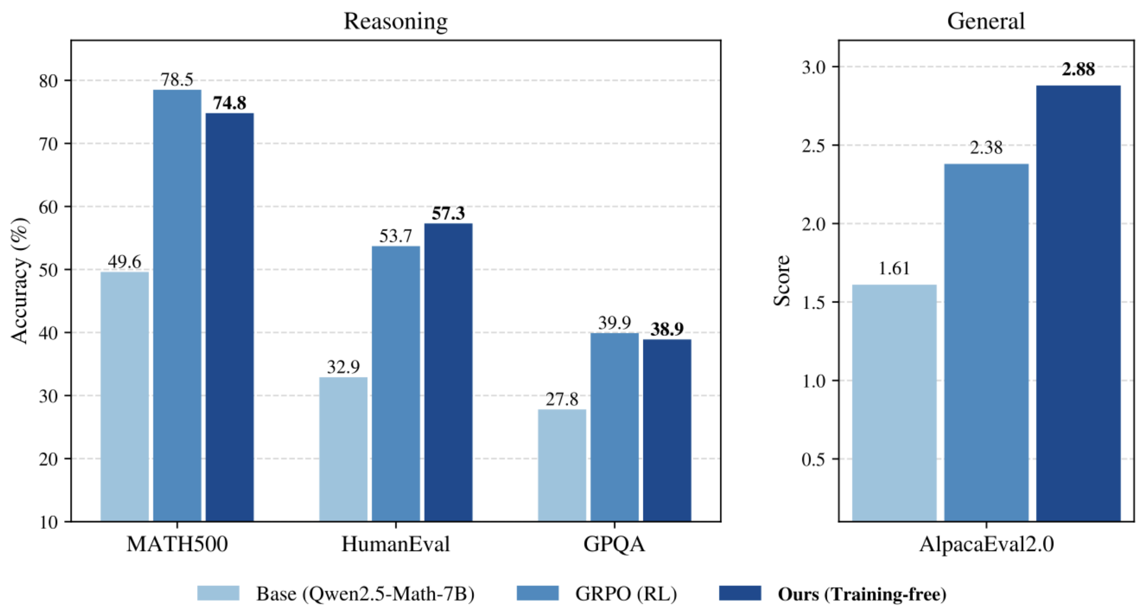

Figure 3: Main headline comparison: base vs GRPO vs power sampling across reasoning tasks and AlpacaEval 2.0.

Table 1 below reports the key numbers reported in the paper (accuracy or score; higher is better).

Table 1: Main results reported in the paper: power sampling vs low-temperature vs GRPO.

Model	**Method**	**MATH500**	**HumanEval**	**GPQA**	**AlpacaEval 2.0**
Qwen2.5-Math-7B	Base	0.496	0.329	0.278	1.61
	Low-temp	0.690	0.512	0.353	2.09
	Power Sampling	**0.748**	**0.573**	**0.389**	**2.88**
	GRPO (MATH)	**0.785**	**0.537**	**0.399**	**2.38**
Qwen2.5-7B	Base	0.498	0.329	0.278	7.05
	Low-temp	0.628	0.524	0.303	5.29
	Power Sampling	**0.706**	**0.622**	**0.318**	**8.59**
	GRPO (MATH)	**0.740**	**0.561**	**0.354**	**7.62**
Phi-3.5-mini-instruct	Base	0.400	0.213	0.273	14.82
	Low-temp	0.478	0.585	0.293	18.15
	Power Sampling	**0.508**	**0.732**	**0.364**	**17.65**
	GRPO (MATH)	**0.406**	**0.134**	**0.359**	**16.74**

Takeaways. On in-domain math (MATH500), power sampling is close to GRPO. On out-of-domain tasks (HumanEval and AlpacaEval), power sampling can outperform GRPO for the evaluated models, suggesting improved generalization beyond the RL training domain.

5.3 Discussion and Analysis

5.3.1 Likelihood/Confidence Concentration and Diversity Collapse

The paper analyzes response log-likelihood under the base model and "confidence" (average negative entropy of next-token distributions):

$$\mathrm{Conf}(x_{0:T}) = \frac{1}{T+1} \sum_{t=0}^{T} \sum_{x \in \mathcal{X}} p(x \mid x_{<t}) \log p(x \mid x_{<t}). \tag{7}$$

Empirically, GRPO responses concentrate at very high likelihood/confidence, while power sampling shifts upward but preserves noticeable spread—consistent with less mode collapse.

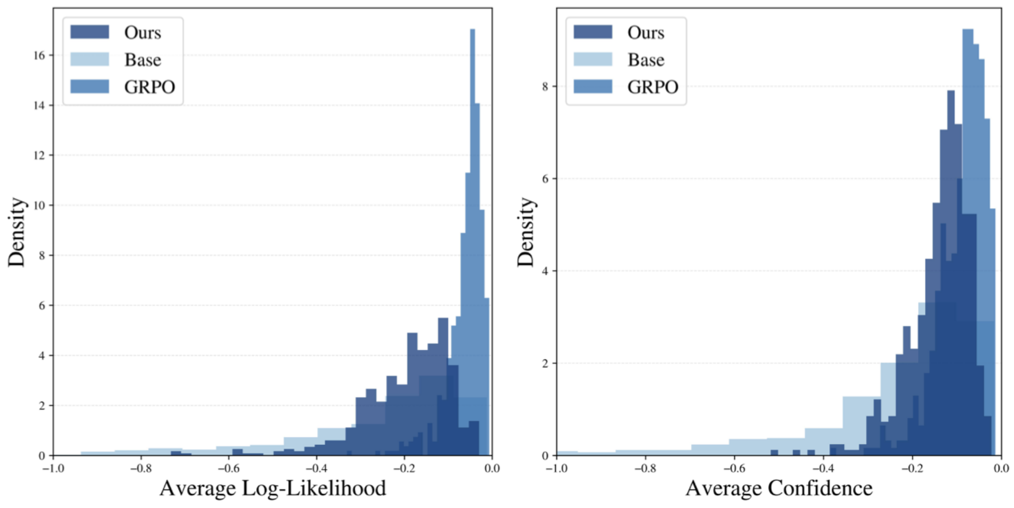

Figure 4: Histograms of base-model log-likelihood and confidence for MATH500 responses: base vs GRPO vs power sampling.

5.3.2 Pass@k: Multi-sample Performance as a Diversity Proxy

To quantify diversity, the paper reports pass@k accuracy: a problem is solved if at least one of k independent samples is correct. Power sampling improves pass@k over GRPO for $k > 1$ and tends to approach the base-model upper bound at larger k, indicating it avoids severe diversity collapse often observed with RL-style post-training method.

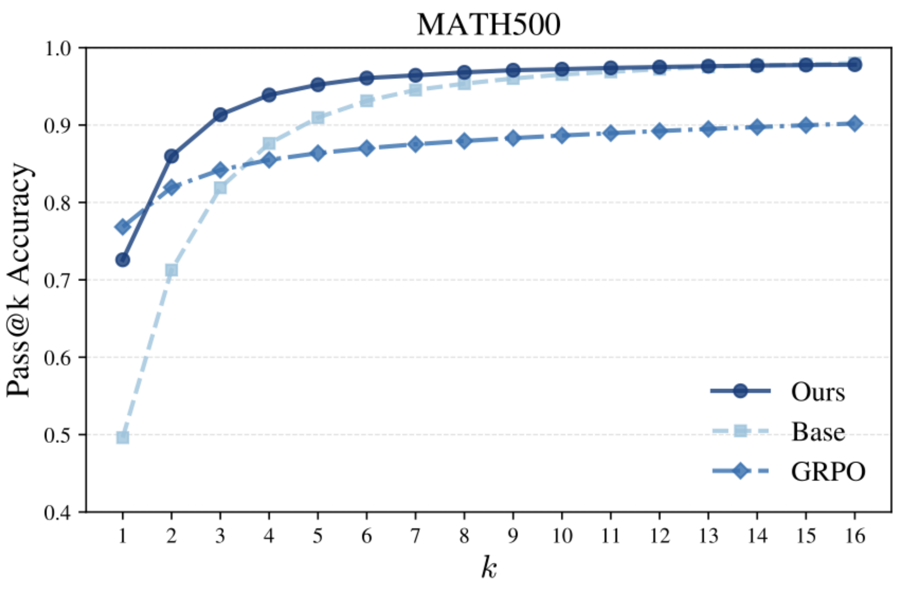

Figure 5: Pass@k on MATH500: power sampling vs base vs GRPO.

5.3.3 Hyperparameters: Effect of α and N_{MCMC}

The paper explores sensitivity to the power α and MH steps N_{MCMC}. It reports a strong performance around $\alpha \approx 4$ and increasing accuracy with N_{MCMC} up to roughly 10 steps.

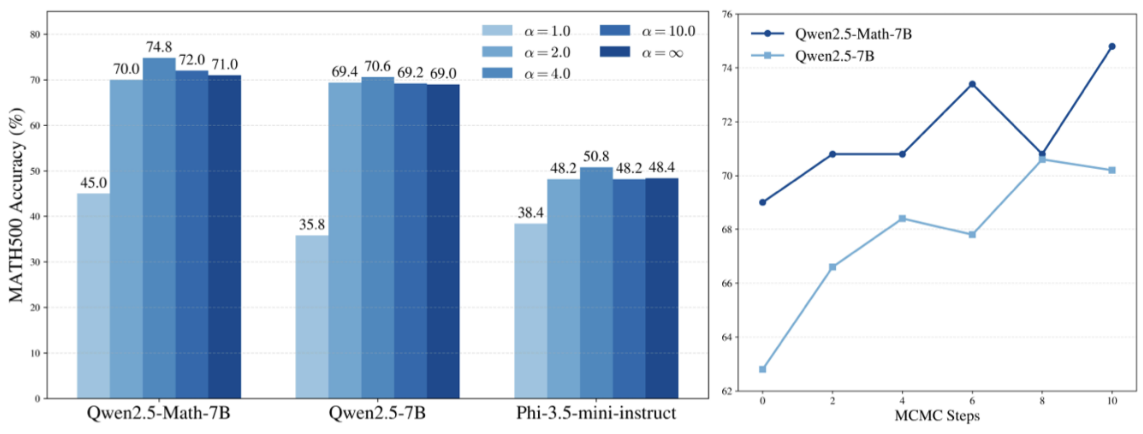

Figure 6: Effect of hyperparameters on MATH500: varying α and N_{MCMC}.

5.3.4 Qualitative Examples

The paper includes examples where power sampling produces correct solutions while GRPO fails on out-of-domain tasks (e.g., some HumanEval prompts). We include one illustrative case in the table below :

Table 2: Qualitative example (from the paper's HumanEval comparison).

Method	**Response**	**Passed**
Power Sampling	`return [s for s in strings if s.startswith(prefix)]`	true
GRPO	`return [string for string in strings if string.startswith(f'{prefix}'*2)]`	false

6 Limitations and Open Questions

- **Compute overhead:** power sampling uses multiple proposal generations and MH steps; it is an inference-time scaling method and can be significantly more expensive than standard decoding.
- **Targeting likelihood vs correctness:** while higher base-model likelihood correlates with correctness on some tasks, it is not guaranteed; the optimal α may vary by domain.
- **Mixing and dependence on proposal:** MH performance depends on proposal quality, block size, and mixing; rigorous bounds are difficult in high-dimensional sequence spaces.
- **Evaluation scope:** the reported gains are strongest on tasks where base models already contain substantial latent competence; it remains to be explored how far the method can go for truly novel skills.

7 Conclusion

This paper argues that base LLMs contain more usable reasoning capability than typical decoding reveals. By explicitly targeting a sharpened distribution p^α and approximating sampling from it using blockwise MH resampling, the proposed "power sampling" method achieves strong single-shot improvements that can rival RL post-training while preserving diversity and boosting pass@k. Conceptually, **it reframes part of "reasoning" progress as an *inference-time sampling problem* rather than an inherently training-time capability creation problem**, and it suggests a promising direction for inference-time scaling in generative models beyond verifiable domains.

References

[1] Mark Chen, Jerry Tworek, Heewoo Jun, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[2] Yann Dubois, Balazs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

[3] Aayush Karan and Yilun Du. Reasoning with sampling: Your base model is smarter than you think. *arXiv preprint arXiv:2510.14901*, 2025.

[4] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[5] David Rein et al. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

A My own experimental reproduction on MATH500

This section reports my own experimental reproduction of the research paper's main math benchmark (MATH500), using the same overall evaluation protocol (single-shot accuracy and pass@k) but with a reduced-scale setup. The motivation is purely computational: power sampling requires many additional forward passes during decoding due to Metropolis–Hastings (MH) resampling, and the full paper configuration (large model, long maximum length, many MCMC steps, and many random seeds) is expensive even on free google colab GPUs.

A.1 Experimental Setup and Methodology

Task. We evaluate on **MATH500**, a standardized subset of the MATH test split, where the model must output a final mathematical answer. We follow the common convention of prompting the model to provide step-by-step reasoning and to place the final answer in `\boxed{...}` for automatic grading.

Model. We use **Qwen/Qwen2.5-Math-1.5B-Instruct**, a smaller model than the research paper's main Qwen2.5-Math-7B results. This choice substantially reduces cost and allows us to iterate and validate the pipeline end-to-end.

Compared inference methods. We compare three decoding strategies:

- **Base sampling**: standard autoregressive sampling from the model.
- **Low-temperature sampling**: sampling from the same model with temperature $\tau = 1/\alpha$ (we use $\alpha = 4 \Rightarrow \tau = 0.25$).
- **Training-free Power Sampling (Power-MH)**: blockwise power sampling implemented with Metropolis–Hastings resampling over blocks of size B, using a low-temperature proposal distribution.

Reduced-scale configuration. To keep runtime manageable, I run only a **subset of 100 problems** from MATH500 and use **3 random seeds** to estimate pass@k. Our key hyperparameters are:

- Maximum generation length: **1024** new tokens for all methods.
- Power Sampling: $\alpha = 4$, block size $B = 192$, and $N_{\textbf{MCMC}} = 3$ MH steps per block.
- pass@k: computed over $k \in \{1, 2, 3\}$ corresponding to our 3 seeds.

This differs from the paper's main setup, which uses a larger model and a larger maximum length ($T_{\text{max}} = 3072$), and typically more MCMC steps (e.g. $N_{\text{MCMC}} = 10$).

Implementation details and reproducibility. We implemented a simple pipeline that (i) generates responses to a standardized CSV schema, (ii) merges all runs into a single artifact, (iii) evaluates accuracy, and (iv) computes pass@k curves. For robustness, we evaluate mathematical equivalence beyond strict string matching by extracting the final answer from `\boxed{...}`.

A.2 Command-line Script Used

Below is the **exact experimental settings** used in my Google Colab run, and the sequence of commands executed.

```
# Task / data
TASK          = "MATH500"
MAX_EXAMPLES = 100   #subset of MATH500 (first 100 problems)
SEEDS         = [0, 1, 2]

# Model
MODEL = "Qwen/Qwen2.5-Math-1.5B-Instruct"

# Decoding / generation
MAX_TOKENS  = 1024    # max_new_tokens

# Low-temperature baseline
ALPHA = 4       # temperature tau=1/alpha

# Training-free Power Sampling (Power-MH)
B = 192    # block size
N_MCMC = 3      # MH steps per block
```

Commands executed : I ran generation for each method (looping over seeds), then evaluated, and finally computed pass@k:

```
python scripts/generate_math500.py --method base (for seed in SEEDS)
python scripts/generate_math500.py --method lowtemp (for seed in SEEDS, alpha=4)
python scripts/generate_math500.py --method power_mh
(for seed in SEEDS, alpha=4, B=192, n_mcmc=3)

python scripts/merge_math500.py
python scripts/evaluate_math500.py
python scripts/passk_math500.py      --max_k 3
```

A.3 Results

Single-shot accuracy (pass@1). Figure 7 summarizes my single-shot accuracies on the 100-problem subset. Power-MH improves over the base model, and also slightly improves over low-temperature sampling.

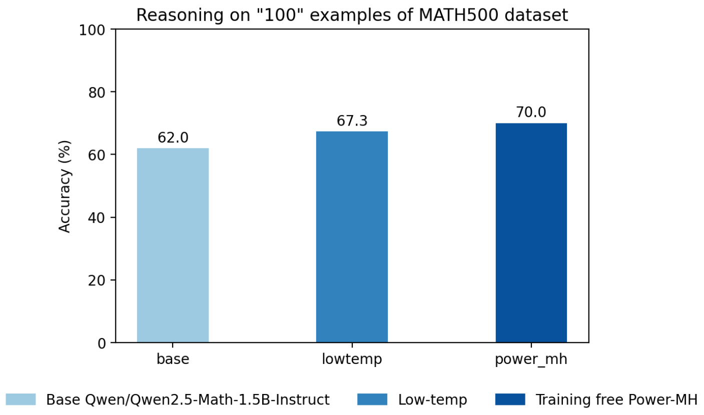

Figure 7: Single-shot accuracy (%) on a 100-example subset of MATH500 for Qwen2.5-Math-1.5B-Instruct: base sampling vs. low-temperature sampling vs. training-free Power-MH.

Table 3 below compares our subset-scale results to the paper's main MATH500 numbers reported for *Qwen2.5-Math-7B*. Note that this comparison is clearly not equivalent due to: (i) a smaller model (1.5B vs 7B), (ii) fewer evaluation examples (100 vs 500), (iii) shorter maximum length (1024 vs 3072), and (iv) fewer MH steps (3 vs 10). These differences are expected to change absolute accuracy and the shape of pass@k curves.

Setting	Base	Low-temp	Power-MH	GRPO
Our run (100 ex, 1.5B)	62.0	67.3	70.0	–
Paper (500 ex, 7B)	49.6	69.0	74.8	78.5

Table 3: MATH500 accuracy comparison: my reduced-scale reproduction vs. paper results.

pass@k (multi-seed): I also compute pass@k over three independent seeds ($k \leq 3$) in Figure 8.

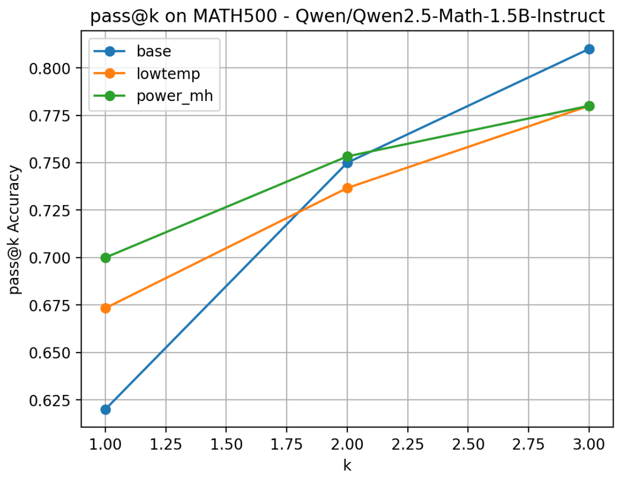

Figure 8: pass@k on a 100-example subset of MATH500 for Qwen2.5-Math-1.5B-Instruct, computed over three seeds ($k \in \{1, 2, 3\}$).

Effect of varying N_{MCMC} (Power-MH) : I performed a small sweep over the number of Metropolis–Hastings steps per block for training-free Power-MH sampling, keeping all other hyperparameters fixed ($\alpha = 4$, $B = 192$, top-$p = 1.0$, max_new_tokens $= 1024$). We used the `Qwen/Qwen2.5-Math-1.5B-Instruct` model, and a 30-problem subset of MATH500 to keep the experiment tractable. Figure 9 shows that increasing N_{MCMC} from 1 to 4 monotonically improves accuracy in our run (from $\approx 66.7\%$ at $N_{\text{MCMC}} = 1$ to $\approx 73.3\%$ at $N_{\text{MCMC}} = 4$), with a brief plateau between $N_{\text{MCMC}} = 2$ and 3. This trend is consistent with the paper's main qualitative finding that additional MH refinement steps increase the quality of power-sampled solutions. However, our sweep is much smaller than the paper's setting (fewer examples, one seed, smaller model), so the curve should be interpreted cautiously: on only 30 problems, the observed plateau and the magnitude of the gain can be strongly affected by sampling variance. A more faithful reproduction would average over multiple seeds and evaluate on the full MATH500 set.

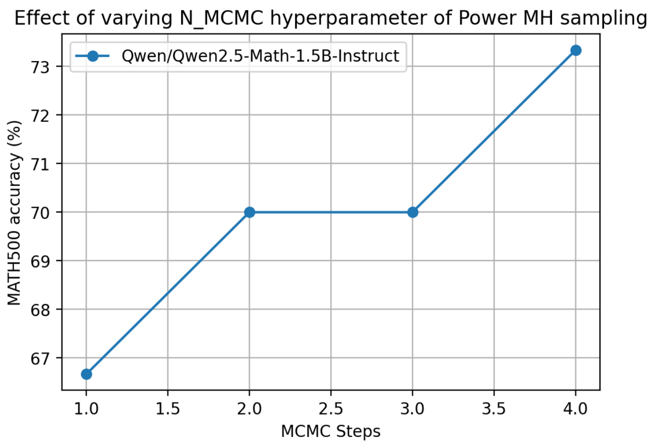

Figure 9: Effect of varying N_{MCMC} of training-free Power-MH sampling on a 30-example subset of MATH500 using `Qwen/Qwen2.5-Math-1.5B-Instruct`, with $\alpha = 4$, $B = 192$, and max_new_tokens $= 1024$.

A.4 Critical Discussion

Overall, our reproduction supports the paper's central empirical claim: **training-free power sampling (Power-MH) can provide a consistent improvement over standard decoding**. On our 100-example subset, Power-MH yields the best single-shot accuracy among the three decoding strategies.

However, our **pass@k behavior** differs from the paper's strongest findings in two important ways. First, our k is small ($k \leq 3$) due to the limited number of seeds; in the paper, curves extend to much larger k (e.g. $k = 16$), where diversity effects become clearer and Power-MH's advantage over RL baselines is most pronounced. Second, the reduced-scale setup introduces noticeably higher variance: with only 100 problems, a few hard or easy items can significantly shift pass@k. For these reasons, the relative ordering of methods at $k = 3$ in our plot should not be over-interpreted.

Finally, the paper emphasizes that Power-MH becomes more effective with increased inference-time compute (e.g. increasing N_{MCMC}) and long generation budgets, while remaining robust to α once $\alpha \gtrsim 2$. In our constrained setting ($N_{\text{MCMC}} = 3$, $T = 1024$), we likely under-utilize the full potential of the method. A natural next step would be to scale N_{MCMC} toward the paper's regime (e.g. 5–10), and/or increase the number of seeds to better estimate pass@k and diversity trends.