THE SURPRISING EFFECTIVENESS OF REASONING WITH UNSURPRISING SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Training language models to reason via reinforcement learning has led to dramatic improvements in their mathematical and coding abilities. Reasoning training leads to models that expend more inference time compute on their chain-of-thought in order to more effectively solve complex problems. Recently it has been shown that it is possible to obtain similar performance to RL training via sampling techniques which only make use of probabilities from the base LLM. In particular, forms of Markov chain Monte Carlo (MCMC) that converge to the annealed LLM distribution yield similar or better performance than RL training at the cost of scaling up inference time compute. In this paper we investigate the power of sampling for reasoning tasks, and discover that MCMC is not necessary to outperform RL. In fact, we show that best-of-N sampling, where the best sequence is determined by the minimum surprisal rate, matches the performance of MCMC sampling methods while being more token efficient and much simpler to implement. Our results demonstrate that the base LLM's level of surprise in its own generated sequences is a strong signal of correctness, allowing for more accurate sampling without any training, external correctness signals, or non-trivial modifications of standard LLM sampling pipelines.

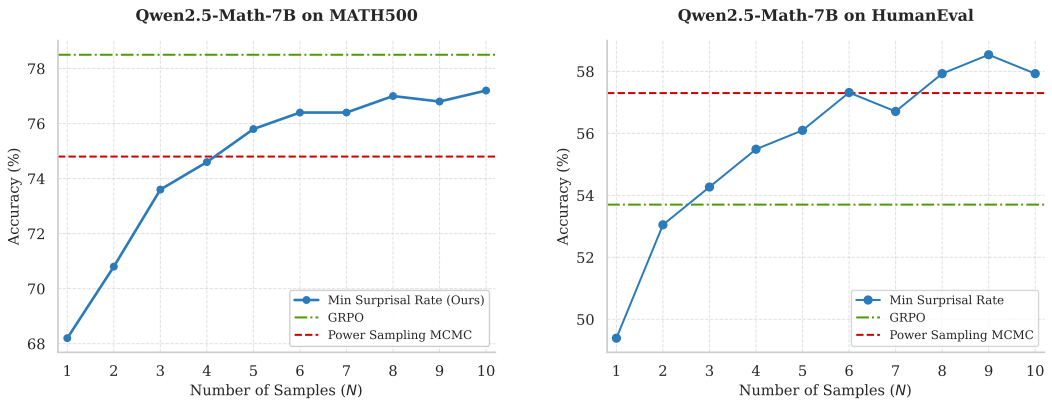
1 INTRODUCTION

The paradigm of reinforcement learning from verifiable rewards (RLVR) has led to breakthroughs in the mathematical, scientific and coding capabilities of LLMs (Jaech et al., 2024; Shao et al., 2024). At the core of this progress is that RLVR training teaches the model to expend additional inference time compute in the form of chain-of-thought reasoning, before outputting its final answer. This additional reasoning effort leads to dramatically improved performance on coding and mathematics benchmarks. At the same time, questions have been raised as to how exactly these benefits arise.

It has been shown that in some cases even RL training with *random rewards* can lead to significant uplift in performance (Shao et al., 2025). Furthermore, standard algorithms such as Group Relative Policy Optimization (GRPO) (Shao et al., 2024) have been shown to produce much of their gains by simply "sharpening" the LLM distribution by putting more probability mass on higher-likelihood completions (Yue et al., 2025). This improves the probability of outputting the correct solution on the first attempt, but can lead to worse Pass@K performance when compared to the base model. Overall, the question of how much the base model already knows, and what the effect of RLVR is on eliciting or expanding this knowledge remains a frontier in research on reasoning in LLMs.

Motivated by these questions, recent work of Karan & Du (2026) examined the possibility of eliciting strong reasoning performance by novel methods of sampling from the LLM. This work considers attempting to sample from the *annealed distribution* of the LLM, which corresponds to reweighting the LLM distribution by raising the probability of each completed sequence to some power $\alpha > 1$. This has the effect of sharpening the distribution, and thus could potentially lead to improved reasoning performance. In general this distribution is intractable to sample from when given access only to the base model. However, Markov chain Monte Carlo (MCMC) methods initially developed for estimating averages in statistical physics are designed precisely for this type of problem. This led the authors to develop the Power Sampling MCMC method, which uses a variant of the Metropolis-Hastings algorithm to attempt to sample from the annealed distribution. Remarkably, Power Sampling MCMC was shown to achieve comparable performance on reasoning tasks to models

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.



(a) MATH 500 Dataset.

(b) HumanEval Dataset.

Figure 1: Accuracy of picking the minimum surprisal rate out of N samples from the base LLM. GRPO is trained on the MATH dataset only. Min surprisal sampling exceeds the accuracy of prior MCMC techniques with only a few samples. Further, it is competitive with GRPO in-distribution on MATH500, and exceeds GRPO out-of-distribution on HumanEval.

trained via the workhorse GRPO algorithm. This is despite the fact that Power Sampling makes no use of verifiable rewards, or any reward signal of any kind, beyond the LLMs own probabilities.

The success of Markov chain Monte Carlo in eliciting strong reasoning performance without rewards raises intriguing further questions. Classically MCMC algorithms require a very large number of iterations to converge to the annealed distribution, so the relatively modest extra sampling cost required for Power Sampling MCMC seems surprisingly low. More fundamentally, these results show that the LLMs own probabilities for token sequences contain a correctness signal that is strong enough to nearly match training with verifiable rewards. However, standard sampling fails to fully take advantage of this signal. Given this fact, the following basic research question arises:

What is the most efficient way to utilize base LLM probabilities to sample correct reasoning?

In this paper we study reasoning via sampling in an attempt to answer this question. Our key finding is that sampling N times from a base LLM and selecting the sample with the *minimum surprisal rate* either matches or exceeds the performance of the Power Sampling MCMC method and GRPO, while consuming significantly fewer tokens. We further investigate this surprising result and make the following contributions:

- We empirically demonstrate the effectiveness of minimum surprisal rate sampling for Qwen2.5-Math-7B, Qwen2.5-7B, and Phi3.5-mini-instruct across mathematical and coding reasoning tasks from the MATH500 and HumanEval datasets. In all cases, minimum surprisal rate sampling matches or outperforms Power Sampling. Furthermore, when compared to GRPO trained on MATH, our method approaches the in-distribution performance of GRPO as measured by MATH500, and exceeds the out-of-distribution performance of GRPO on HumanEval.
- We provide theoretical justification for the performance of independent sampling for finding a high-probability mode of a distribution on sequences. In particular, we prove a no-free-lunch theorem for sampling a mode–without assuming additional structure on the LLM distribution, every sampling algorithm that only has access to probabilities from the base model will not have significantly higher chance of sampling a mode than minimum surprisal sampling.
- Motivated by avoiding our no-free-lunch result, we investigate if there are any additional structural properties present in LLM distributions through the lens of the surprisal rate. We find that measuring the surprisal rate of prefixes of the full sequence can in fact lead to a more accurate correctness signal, and improved performance for minimum surprisal sampling.

2 Related Work

The work of Karan & Du (2026) introduced Power Sampling MCMC to draw samples from the annealed LLM distribution, and demonstrated its effectiveness relative to GRPO. While the additional cost in tokens is approximately $9\times$ standard sampling on average, the latency is much larger due to the serial nature of the Metropolis-Hastings algorithm. Follow-up work on the use of Markov chain Monte Carlo for reasoning with sampling has focused on the use of parallelization via Sequential Monte Carlo (Azizi et al., 2026) or limited look-ahead (Ji et al., 2026) in order to reduce the latency of Power Sampling. In contrast to this prior work, our sampling method uses many fewer tokens, and is immediately compatible with existing optimized LLM batch sampling pipelines, enabling highly efficient sampling via reasoning.

There has been a significant body of research on the degree to which RLVR achieves its performance by sharpening the original LLM distribution, going back to the original paper that introduced GRPO (Shao et al., 2024). This has been viewed as a limitation of the ability to utilize the verifiable reward effectively in some cases (Shao et al., 2025), leading to degraded Pass@K performance. Further research has shown evidence that the quality of LLM reasoning does in fact improve during RL training as measured by chain-of-thought correctness metrics (Wen et al., 2025). There has also been research to train Pass@K directly, in-order to avoid collapse in reasoning diversity (Walder & Karkhanis, 2026).

Research on self-improvement via RL training attempts to leverage the intuition that the base LLM might be able to accurately judge correctness more effectively than it can actually generate correct solutions (Pang et al., 2024; Yuan et al., 2024). Interestingly, work in this direction has also identified sharpening of the LLM distribution as a key contributor to performance (Huang et al., 2024). These methods relate to research on reasoning with sampling, in the sense that they utilize the base LLM to provide reward, but are quite distinct because they still rely on computationally intensive RL training.

Guiding LLM sampling with an external reward signal has received significant attention as a method to improve performance at test time (Zhao et al., 2024; Lew et al., 2023; Feng et al., 2024; Loula et al., 2025). In this area a prominent approach is Sequential Monte Carlo or particle filtering methods. Generally, these methods auto-regressively rollout N samples in parallel, estimate rewards with a process-feedback verifier, then re-weight and resample the N rollouts. The resampling process implicitly prunes rollouts that are doing poorly according to the verifier, while allowing multiple possible completion attempts for rollouts that are doing well. While these techniques are inference-only, they rely on an external trained process reward signal, while reasoning with sampling uses only the base LLM probabilities.

3 Background and Preliminaries

We begin by introducing notation that will be used throughout the paper. Let $\mathcal{X}$ be a finite set of tokens, and let $p(\cdot)$ be a probability distribution on sequences of at most T tokens $x = x_1, x_2, \dots x_l$ for $l \leq T$, where each $x_t \in \mathcal{X}$. We will use the notation $x_{<t} = x_1, x_2, \dots, x_{t-1}$ to denote the length $t-1$ prefix of x. We write $p(\cdot \mid x_{<t})$ for the distribution of p conditioned on the prefix $x_{<t}$. We will use $|x|$ to denote the length of the sequence x.

3.1 Surprisal

For a distribution p the *surprisal* of a sequence x is given by $-\log p(x)$. The entropy of the sequence is the expectation of the surprisal. Clearly, lower values of the surprisal for x correspond directly to larger values of the probability $p(x)$. Further, the surprisal of a sequence is just the sum of the conditional surprisals

$$-\log p(x) = -\log\left(\prod_t p(x_t \mid x_{<t})\right) = -\sum_t \log p(x_t \mid x_{<t}).$$

Hence, for many distributions of interest–such as LLMs–where each conditional distribution has non-trivial entropy, the surprisal tends to increase with sequence length. This means that longer sequences will have lower surprisal, simply due to their length. Thus, to more accurately capture

"how surprising" a sequence is, we normalize by the length of the sampled sequence. We formalize this in the following definition.

Definition 3.1 (Surprisal Rate)**.** For a distribution p and a sequence x, the *surprisal rate* of x with respect to p is given by

$$\mathrm{SR}_p(x) = -\frac{1}{|x|}\log p(x).$$

3.2 Power Sampling via MCMC

A recent approach to reasoning via sampling has focused on sampling from the *annealed distribution*. Given a probability distribution on sequences $p(x)$ and $\alpha > 1.0$ the corresponding annealed distribution is $p_\alpha(x) \propto p(x)^\alpha$. Formally we have

$$p_\alpha(x) = \frac{p(x)^\alpha}{\sum_{x'} p(x')^\alpha}. \tag{1}$$

In other words, the original distribution p is reweighted by raising each probability to the power α. This has the effect of *sharpening* the distribution i.e. it increases the probability mass on higher probability sequences relative lower probability sequences under $p(x)$.

It has been widely hypothesized that RL math and code reasoning training for LLMs works by sharpening the LLM distribution. That is, the LLM in a sense "already knows" how to solve many reasoning problems, in the sense that the high-probability modes of the distribution are centered on correct reasoning, but there is enough entropy that just by chance the model will often make a mistake. Reinforcement learning from verifiable rewards is thought to sharpen the distribution, so that more of the mass is concentrated on the correct reasoning modes. Hence, it makes sense to attempt to directly sample from the annealed distribution, which has a similar sharpening effect on the correct reasoning modes.

Crucially, sampling from the annealed distribution p_α is not equivalent to sampling with low temperature $T = \frac{1}{\alpha}$. To see this observe that the conditional annealed distribution is proportional to

$$p_\alpha(x_t \mid x_{<t}) \propto \sum_{x_{>t}} p(x_{<t}, x_{>t})^\alpha \tag{2}$$

while the conditional low-temperature distribution is proportional to

$$p_{\text{low-temp}}(x_t \mid x_{<t}) \propto \left(\sum_{x_{>t}} p(x_{<t}, x_{>t})\right)^\alpha. \tag{3}$$

In general, when $\alpha > 1.0$ the sum of exponents will not be equal to the exponent of the sum, hence these two distributions are unequal.

While it is straightforward to sample from the low-temperature distribution, it is not at all obvious how best to sample from the annealed distribution. As the name might suggest, this task is a longstanding computational challenge in statistical physics and probabilistic modelling, and the classical Metropolis-Hastings algorithm gives a generic Markov chain monte carlo method to achieve it. To summarize briefly, in each iteration of Metropolis-Hastings there is a current sample x. Then a new sample is y is drawn from a *proposal distribution* $q(y \mid x)$. Finally, with probability

$$A(x, y) = \min\left\{\frac{p^\alpha(y)q(x \mid y)}{p^\alpha(x)q(y \mid x)}, 1\right\} \tag{4}$$

the proposed sample y is selected for the next iteration, and otherwise x remains the sample used in the next iteration. The most important point is that one just needs to compute ratios of probabilities p^α, which is tractable given only access to p, because the intractable sum in the denominators cancel out. This markov chain is known to converge to the true annealed distribution p_α under mild assumptions on the proposal distribution q.

Power Sampling MCMC is an approach to sampling the annealed distribution of an LLM based on the Metropolis-Hastings algorithm. In this case, the proposal distribution $q(y \mid x)$ is given by picking a uniform random index t and resampling from the conditional low-temperature distribution

$y \sim p_{\text{low-temp}}(\cdot \mid x_{<t})$. Metroplis-Hastings and related MCMC methods typically take many steps to converge to annealed distribution. To make this more practical with LLMs, Power Sampling MCMC generates a block of length B tokens at a time. So first Metropolis-Hastings is run for N steps to sample $p_\alpha(x_{<B})$, then for N more steps to sample from $p_\alpha(x_{<2B})$ and so on until the stop token is reached.

4 Sampling Unsurprising Sequences

As described in the previous section, the prior work on sampling from the annealed distribution is justified by the idea that the modes of the distribution are likely to correspond to correct solutions. Thus, sharpening the distribution to its annealed version will put more probability mass on modes corresponding to correct samples, yielding better performance. While this idea and explanation make intuitive sense, it is not immediately clear that this is the most efficient way to focus the sampling onto the high-probability modes of the distribution $p(\cdot)$.

In fact, given that we have access to the actual log-probabilities $p(x)$ for any input x, a sensible alternative might be to just draw N samples and pick the one with the highest probability i.e. the minimum surprisal. The only issue with this idea is that LLMs generate sequences of varying length, and so longer sequences tend to have lower probability (higher surprisal) due to their length alone. To adjust for this issue we can instead draw N samples and pick the one with the *minimum surprisal rate* as defined in Definition 3.1. This leads to an intuitively appealing Best-of-N algorithm, where we just select the sequence that the LLM found least surprising *per-token*. The full pseudocode is provided in Algorithm 1.

Algorithm 1 Min Surprisal Rate Sampling

Require: Language model distribution $p(x)$, Sample size N
Ensure: The minimum surprisal rate sample x^*
1: $\mathcal{S} \leftarrow \emptyset$
2: **for** $i = 1$ **to** N in parallel **do**
3: $\quad x^{(i)} \sim p(x)$
4: $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{x(i)\}$
5: **end for**
6: $x^* \leftarrow \arg\min_{x \in \mathcal{S}} \mathrm{SR}_p(x) = \arg\min_{x \in \mathcal{S}} -\frac{1}{|x|} \log p(x)$
7: **return** x^*

4.1 No Free Lunch for Sampling a Mode

While it may initially seem naive to expect best-of-N sampling to match the performance of more complex MCMC methods, there are principled reasons to expect that it might. As a starting point, classical MCMC methods used to estimate properties of the annealed distribution in statistical physics typically require a very large number of samples in order to converge. This makes the strong performance of Power Sampling MCMC quite surprising, because it only uses $N = 10$ MCMC steps per round. Thus, it is reasonable to wonder if the good performance is happening for some different reason, rather than convergence of MCMC to the true annealed distribution p_α. In fact, the Metropolis-Hastings algorithm applied to p_α clearly favors resampling sequences that have higher probability under p. Therefore one plausible explanation for the performance is that higher probability sequences are more likely to be correct, and Power Sampling MCMC favors higher probability sequences.

In the remainder of this section, we will give a stronger theoretical motivation for the benefits of Min Surprisal Rate sampling, in the form of a "no free lunch" theorem for sampling a high-probability mode of a distribution $p(\cdot)$. Formally, we will show that any sampling algorithm that is restricted to only querying probabilities $p(x)$ will not outperform min surprisal rate sampling on some distributions. We begin by formalizing the task of sampling a high-probability mode. First, we define $\mathcal{P}(\beta)$ to be the set of probability distributions p on sequences of length exactly T, such that there is a mode that is sampled with probability at least β. Formally, for $\beta \in [0, 1]$ we define

$$\mathcal{P}(\beta) = \{p(\cdot) \mid \exists x^* \text{ such that } p(x) > 0 \implies |x| = T \text{ and } p(x^*) > \beta\} \tag{5}$$

We next turn to the class of sampling algorithms to which our no free lunch theorem will apply.

Definition 4.1. A Q-query *mode sampling algorithm* $\mathcal{A}$ is a randomized algorithm that makes Q adaptive queries of the form $p(x^{(i)})$ for $i \in \{1, \ldots, Q\}$ to a distribution $p \in cP(\beta)$, and then outputs a sample x. The algorithm succeeds if it outputs x^* satisfying $p(x^*) > \beta$ and otherwise the algorithm fails.

Clearly min surprisal rate sampling with N samples is a Q-query mode sampling algorithm. We now consider what happens when we apply min surprisal rate sampling to any distribution in $p \in \mathcal{P}$.

Proposition 4.1 (Min Suprisal Rate Performance)**.** *For any natural number N and distribution $p \in \mathcal{P}$ min surprisal rate sampling fails with probability*

$$P_{\textit{min-surprisal}} \leq (1-\beta)^N$$

Proof. Because all samples in the support of p have the same length T, if one of the N samples has $p(x) > \beta$, then min surprisal rate sampling will output a sample x^* with $p(x^*) > \beta$. The fact that $p \in \mathcal{P}$ implies that there is at least one sequence x^* with probability $p(x^*) > \beta$. Therefore, the probability that an individual sample $x \sim p(\cdot)$ has $p(x) \leq \beta$ is at most $1 - \beta$. Since the N samples are independent, the probability of failure i.e. that all samples have $p(x) \leq \beta$ is at most

$$(1-\beta)^N.$$

□

We now turn to our no free lunch theorem. The central point is that general distributions $p \in \mathcal{P}(\beta)$ can be completely unstructured, so that the only way to find the high-probability mode x^* is blind sampling.

Theorem 4.1 (No Free Lunch for Sampling a Mode)**.** *For any N-query mode sampling algorithm $\mathcal{A}$ and any power of two $\beta = 2^{-k}$, there exists a distribution p such that the probability that $\mathcal{A}$ fails is at least $1 - N\beta - 2^{-T}$.*

Proof. Assume that the set of tokens is $\mathcal{X} = \{0, 1\}$. The proof relies on Yao's Minimax principle. Yao's principle states that if there is a worst-case distribution on inputs such that every deterministic algorithm $\mathcal{D}$ fails with probability at least P, then for every randomized algorithm $\mathcal{A}$ for the same task, there exists a worst-case input such that $\mathcal{A}$ fails with probability at most P. In our setting, the set of possible inputs is the set of distributions $p \in \mathcal{P}(\beta)$. Hence, to prove the theorem it suffices to find a distribution μ on the set of distributions $\mathcal{P}(\beta)$, such that every deterministic query algorithm fails with probability at least $1 - \beta N$.

Recall that $\beta = 2^{-k}$. We define μ as follows. For $z \in \{0, 1\}^T$ let $p_z(\cdot)$ be the distribution given by

$$p_z(x) = \begin{cases} 2^{-T} & \text{if } x_{\leq k} \neq z_{\leq k} \\ 0 & \text{if } x_{\leq k} = z_{\leq k} \text{ and } x_{>k} \neq z_{>k} \\ 2^{-k} & \text{if } x = z \end{cases} \tag{6}$$

In other words, $p(\cdot)$ is the uniform distribution, except that all of the probability mass that would go towards x that agree with z on the first k bits is pushed onto z. By definition we have $p \in \mathcal{P}(\beta)$. Next let $\mathcal{D}$ be any *deterministic* N-query sampling algorithm. Consider the sequence of queries $x^{(1)}, \ldots x^{(N)}$ that $\mathcal{D}$ makes when each probability $p(x^{(i)}) = 2^{-T}$. Because $\mathcal{D}$ is deterministic, this is a single well-defined sequence of N queries.

Now consider the distribution μ on $\mathcal{P}$ that samples a uniform random $z \in \{0, 1\}^T$ and then outputs the distribution $p_z(\cdot)$. The N queries $x^{(1)}, \ldots x^{(N)}$ have at most N different k-bit prefixes, out of a total of 2^k possible k-bit prefixes. Hence, the probability over $p_z \sim \mu$ that $p_z(x^{(i)}) = 2^{-T}$ for all i is at least

$$\Pr[z_{\leq k} \neq x^{(i)}_{\leq k} \; \forall i] \geq 1 - N/2^k = 1 - \beta N \tag{7}$$

Let $\hat{x}$ be the (deterministic) output of $\mathcal{D}$ in the case that all N queries take the value 2^{-T}. The algorithm $\mathcal{D}$ clearly fails if both $z \neq \hat{x}$ and $z_{\leq k} \neq x^{(i)}_{\leq k}\ \forall i$ hold simultaneously. Let E be this event, then by Equation (7)

$$\Pr[E] \geq 1 - \beta N - 2^{-T}. \tag{8}$$

Finally, applying Yao's minimax principle, we conclude that for every randomized sampling algorithm $\mathcal{A}$ there exists a distribution $p(\cdot)$ such that $\mathcal{A}$ fails with probability at least $1 - \beta N - 2^{-T}$. □

When $\beta N < 1$, the upper bound on the failure rate of the min surprisal algorithm in Proposition 4.1 to the lower bound for any N-query sampler in Theorem 4.1 can be quite close. This follows from the second order approximation

$$(1-\beta)^N \leq 1 - N\beta + N(N-1)/2\beta^2 \tag{9}$$

Hence we have that the gap between the upper bound and the lower bound is at most

$$1 - N\beta + N(N-1)/2\beta^2 - 1 + N\beta + 2^{-T} \leq \frac{1}{2}N^2\beta^2 + 2^{-T}. \tag{10}$$

For large values of T the dominant term is $\frac{1}{2}N^2\beta^2$ which decreases at a quadratic rate as βN gets smaller. In other words, the upper and lower bounds match to first order in βN.

Taking a step back, we have shown that in there is no free lunch for mode-sampling algorithms–there will always be a worst-case unstructured distribution, where one cannot do too much better than just drawing N samples and hoping to hit the mode. Of course the question remains if there is some common extra structure in LLM distributions $p(\cdot)$ that allow them to outperform the lower bound. Alternatively, our no free lunch theorem does not apply to settings like particle-filtering or Sequential Monte Carlo methods (Lew et al., 2023; Zhao et al., 2024; Feng et al., 2024), where a process reward verifier provides extra structure to guide sampling. Hence, the power of MCMC techniques to improve performance when an external reward is available is not affected by our results, and one would expect better performance in such settings.

In the subsequent sections we will demonstrate that existing MCMC techniques for reasoning with sampling generally do not perform better than drawing N samples and picking the one with the minimum surprisal rate. We will then further investigate if there is any extra structure that can be exploited, through the lens of the surprisal rate of sampled sequences.

5 Experimental Results

In this section we give our main experimental results, demonstrating that min surprisal rate sampling is competitive with Power Sampling MCMC, while fewer tokens and being implementable in standard LLM inference pipeline. We run our experiments on three standard open weight LLMs Qwen2.5-7B-Math, Qwen2.5-7B, and Phi-3.5-mini-instruct. For measuring mathematical reasoning we used MATH500 (Lightman et al., 2023), which is a randomly sampled subset of 500 questions from the original test subset of the MATH dataset of Hendrycks et al.. The MATH dataset contains 12500 total competition math problems from diverse areas. For measuring coding capability we used the HumanEval dataset (Chen et al., 2021). HumanEval consists of 164 competition coding problems each with several unit tests to verify the correctness of LLM solutions via code execution. For the GRPO baseline we used the implementation from Shao et al. (2025). All experiments were run on a system with an Nvidia RTX4090 GPU, 64 GB of RAM, and an AMD Ryzen 9 7950X 16-Core Processor.

We used the original Power Sampling MCMC, which uses $N = 10$ MCMC steps per round, sets $\alpha = 4.0$ for the target annealed distribution p_α, and uses low-temperature sampling with temperature $1/\alpha = 0.25$ for the proposal distribution. For our min surprisal sampling algorithm we also used low-temperature sampling with the same value of 0.25 as used by the proposal distribution in Power Sampling MCMC.

The performance of min surprisal sampling across a range of values of N is reported in Figure 1. With only $N = 4$ samples, min surprisal sampling matches the performance of Power Sampling MCMC. With a total of $N = 10$ samples our algorithm comes within approximately 1 percentage point of the performance of GRPO trained on the MATH training set. In HumanEval, min surprisal

sampling with $N = 6$ samples matches the performance of Power Sampling MCMC, while larger values of N up to 10 marginally exceed this performance level. Notably, min surprisal sampling outperforms out-of-distribution GRPO i.e. GRPO was trained on the MATH dataset, and then tested on HumanEval.

Table 1 contains the full details of our results across all models and datasets. In general, min surprisal sampling offers comparable or better performance to PowerSampling in all cases, while using many fewer tokens, and being compatible with any high-performance LLM sampling pipeline.

	Qwen2.5-Math-7B		Qwen2.5-7B		Phi-3.5-mini-inst.	
Method	MATH500	HumanEval	MATH500	HumanEval	MATH500	HumanEval
Base	0.496	0.329	0.498	0.329	0.400	0.213
Low-temperature	0.690	0.512	0.628	0.524	0.478	0.585
Power Sampling	0.748	0.573	0.706	0.622	0.508	0.732
Min Surprisal (ours)	**0.772**	**0.579**	**0.698**	**0.634**	**0.496**	**0.713**
GRPO (MATH)	0.785	0.537	0.740	0.561	0.406	0.134

Table 1: Comparison of min surprisal sampling, Power Sampling, and GRPO (trained on MATH) across models and datasets.

6 Investigating the Surprisal Rate

In this section we study the structure of LLM distributions through the lens of the surprisal rate. The overall goal is to understand if there is non-trivial structure that can be exploited to get around Theorem 4.1, our no free lunch theorem for mode sampling. For example, one quite optimistic hypothesis is that the surprisal rate can be used to identify the best sample early on in the token sequence. That is we might instead try to select the sample x minimizing $\mathrm{SR}_p(x_{<t})$ for some $t < T$. If this turns out to provide a strong correctness signal, then we could save on tokens by stopping the sampling for all but a few of the lowest surprisal prefixes.

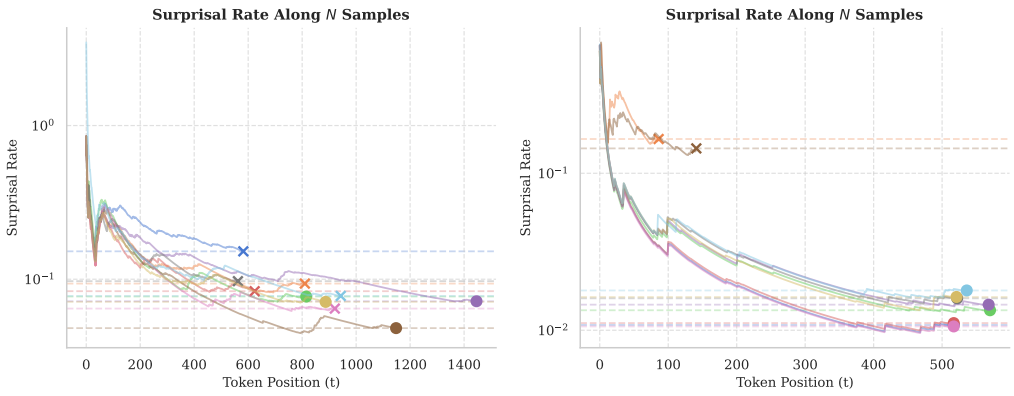


Figure 2: Surprisal rate of Qwen2.5-Math-7B at each token position along N samples for two randomly selected sets of samples. The dotted horizontal lines represent the final surprisal value for each sequence. A circle indicates a correct final answer, a $\times$ indicates incorrect.

We begin with qualitative analysis, by plotting in Figure 2 the surprisal rate across token positions for two randomly selected sets of $N = 10$ samples drawn from Qwen2.5-Math-7B on the MATH500 dataset. On the right-hand facet, there is a strong split between the correct and incorrect samples even after 50 tokens, with two incorrect samples with very high surprisal rates, and eight correct samples with much lower surprisal rates. In this case, stopping sampling for all but the minimum surprisal rate at 50 tokens would have selected the best answer. The left-hand facet shows the selected correct sampled solution (brown) that only beats out an incorrect sampled solution (pink) after nearly 500 tokens. This is nearly half of the length of the final generated sequence. These results illustrate that while in some cases it is surprisal rate offers a clean separation of correct and incorrect samples, in

others one has to wait for a large fraction of the total tokens in order to identify the correct solution via surprisal rate.

To make things more quantitative we now perform a more systematic experiment. For t in the range 0 to 3000 with increments of size 200 we apply the min surprisal algorithm to the full data set of prefixes of length t. That is, for each set of N samples we select the minimum according to $\mathrm{SR}_p(x_{<t})$ and then plot the resulting accuracy as a function of t in Figure 3.

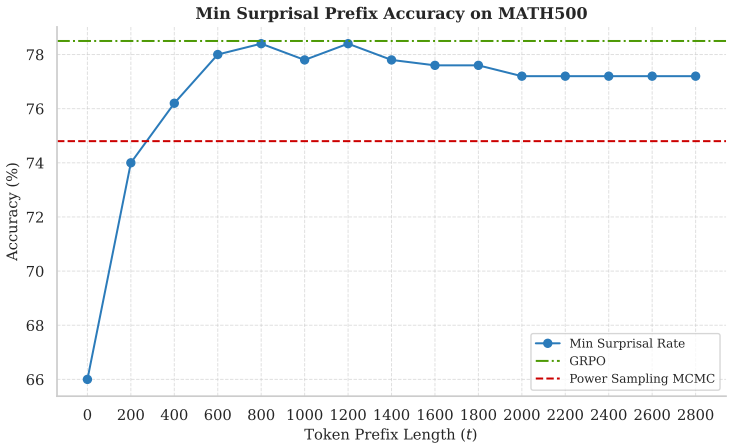


Figure 3: Accuracy of Qwen2.5-Math-7B on MATH500 when calculating the minimum surprisal rate on prefixes of the full sequence. If a sequence terminates before the prescribed prefix length, we use the final surprisal rate value.

The prefix accuracy reveals an interesting structure in the LLM distribution. The correctness signal from the surprisal rate of a prefix increases monotonically up to 800 tokens, and then slowly decreases to its final value. In other words, the algorithm can achieve higher accuracy by selecting the sample with minimum surprisal rate after 800 tokens have been sampled, rather than waiting until the end of each generated sequence. In fact, the performance of min surprisal rate sampling at 800 token prefixes matches that of GRPO in-distribution on the MATH500 dataset. It is possible that these results indicate an "overthinking" phenomenon, where very long sequences indicate that the model has gotten stuck in an overthinking loop, and is actually less likely to be correct. Whether or not this particular hypothesis is true, the results in Figure 3 demonstrate some non-trivial structure in the LLM distribution that could potentially be taken advantage of to further improve reasoning-with-sampling performance. We leave to future work the question of how such improvements can be made.

7 Discussion

We have demonstrated that selecting the minimum surprisal rate out of N samples leads to reasoning performance that matches or exceeds more complex MCMC techniques targeting the annealed distribution. We further theoretically justify this advantage, by proving a no-free-lunch theorem for sampling a high-probability mode of a distribution, demonstrating that it is not possible to significantly outperform min surprisal rate sampling in the worst case for unstructured distributions. Finally, we investigated the structure present in base LLM distributions, and determined that min surprisal sampling on prefixes can offer stronger performance. A limitation of our experimental results is that, due to computational constraints, we were only able to test min surprisal sampling for 7 billion parameter models. The question of how this scales to large, more powerful models remains open, and presents an interesting direction for future research. Another important point is that our no free lunch theorem does not apply to SMC or partical-filtering methods that make use of an external process-supervision reward (Lew et al., 2023; Zhao et al., 2024). Hence, our results may be taken as evidence that process-supervision reward models present a compelling way to leverage additional structure for reasoning via sampling, at the cost of training a process-supervision model.

AI USE STATEMENT

This paper does not contain any content generated by AI.

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. *arXiv preprint arXiv:2602.10273*, 2026.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Shengyu Feng, Xiang Kong, Shuang Ma, Aonan Zhang, Dong Yin, Chong Wang, Ruoming Pang, and Yiming Yang. Step-by-step reasoning for math problems via twisted sequential monte carlo. *arXiv preprint arXiv:2410.01920*, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

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. *arXiv preprint arXiv:2412.01951*, 2024.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Xiaotong Ji, Rasul Tutunov, Matthieu Zimmer, and Haitham Bou Ammar. Scalable power sampling: Unlocking efficient, training-free reasoning for llms via distribution sharpening. *arXiv preprint arXiv:2601.21590*, 2026.

Aayush Karan and Yilun Du. Reasoning with sampling: Your base model is smarter than you think. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Vsgq2ldr4K.

Alexander K Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash K Mansinghka. Sequential monte carlo steering of large language models using probabilistic programs. *arXiv preprint arXiv:2306.03081*, 2023.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The twelfth international conference on learning representations*, 2023.

João Loula, Benjamin LeBrun, Li Du, Ben Lipkin, Clemente Pasti, Gabriel Grand, Tianyu Liu, Yahya Emara, Marjorie Freedman, Jason Eisner, et al. Syntactic and semantic control of large language models via sequential monte carlo. *arXiv preprint arXiv:2504.13139*, 2025.

Jing-Cheng Pang, Pengyuan Wang, Kaiyuan Li, Xiong-Hui Chen, Jiacheng Xu, Zongzhang Zhang, and Yang Yu. Language model self-improvement by reinforcement learning contemplation. In *The Twelfth International Conference on Learning Representations*, 2024.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, et al. Spurious rewards: Rethinking training signals in rlvr. *arXiv preprint arXiv:2506.10947*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Christian Walder and Deep Tejas Karkhanis. Pass@k policy optimization: Solving harder reinforcement learning problems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=W6WC6047X2.

Xumeng Wen, Zihan Liu, Shun Zheng, Shengyu Ye, Zhirong Wu, Yang Wang, Zhijian Xu, Xiao Liang, Junjie Li, Ziming Miao, et al. Reinforcement learning with verifiable rewards implicitly incentivizes correct reasoning in base llms. *arXiv preprint arXiv:2506.14245*, 2025.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-rewarding language models. In *International Conference on Machine Learning*, pp. 57905–57923. PMLR, 2024.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Baker Grosse. Probabilistic inference in language models via twisted sequential monte carlo. In *International Conference on Machine Learning*, pp. 60704–60748. PMLR, 2024.