REASONING WITHOUT TRAINING: YOUR BASE MODEL IS SMARTER THAN YOU THINK

Anonymous authors
Paper under double-blind review

ABSTRACT

Frontier reasoning models have exhibited incredible capabilities across a wide array of disciplines, driven by posttraining large language models (LLMs) with reinforcement learning (RL). However, despite the widespread success of this paradigm, much of the literature has been devoted to disentangling truly novel behaviors that emerge during RL but are not present in the base models. In our work, we approach this question from a different angle, instead asking whether comparable reasoning capabilites can be elicited from base models at inference time, *without any additional training*. Inspired by Markov chain Monte Carlo (MCMC) techniques for sampling from sharpened distributions, we propose a simple iterative sampling algorithm leveraging the base models' own likelihoods. Over different base models, we show that our algorithm offers substantial boosts in reasoning that nearly match and can even outperform those from RL on a wide variety of single-shot tasks, including MATH500, HumanEval, and GPQA. Crucially, our method does not require training, curated datasets, or a verifier, suggesting a general applicability beyond easily verifiable domains.

1 INTRODUCTION

Reinforcement learning (RL) has become the dominant paradigm for enhancing the reasoning capabilities of large language models (LLMs) Guo et al. (2025); Hu et al. (2025). Equipped with a reward signal that is typically automatically verifiable, popular RL techniques have been successfully applied to posttrain frontier models, leading to sizeable performance gains in domains like math, coding, and science Hendrycks et al. (2021); Li et al. (2022); Rein et al. (2024).

Despite the widespread empirical success of RL for LLMs, a large body of literature has centered around the following question: are the capabilities that emerge during RL posttraining *fundamentally novel behaviors* that are *not* present in the base models? This is the question of *distribution sharpening* He et al. (2025); Shao et al. (2025); Yue et al. (2025): that is, whether the posttrained distribution is simply a "sharper" version of the base model distribution, instead of placing mass on reasoning traces the base model is unlikely to generate.

Several works point towards the difficulty in learning new capabilities with RL posttraining. He et al. (2025); Song et al. (2025) compare the pass@k scores of base models with posttrained models, finding that for large k, base models actually outperform while the latter suffer from degraded generation diversity. Yue et al. (2025) also notes that the reasoning traces post-RL have higher likelihoods under the base model, seemingly drawing from existing high likelihood capabilities. Regardless, the advantage of RL posttraining for single-shot reasoning has remained, as of yet, undeniable.

In this paper, we present a surprising result: *sampling directly from the base model can achieve single-shot reasoning capabilites on par with those from RL.*

We propose a sampling algorithm for base models that leverages additional compute at inference time, achieving single-shot performance that *nearly matches* RL posttraining on *in-domain* reasoning tasks and can even *outperform* on *out-of-domain* reasoning tasks. We benchmark specifically against Group Relative Policy Optimization (GRPO), which is the standard RL algorithm for enhancing LLM reasoning Shao et al. (2024).

Crucially, our algorithm is *training-free*, *dataset-free*, and *verifier-free*, avoiding some of the inherent weaknesses of RL methods including extensive hyperparameter sweeps to avoid training insta-

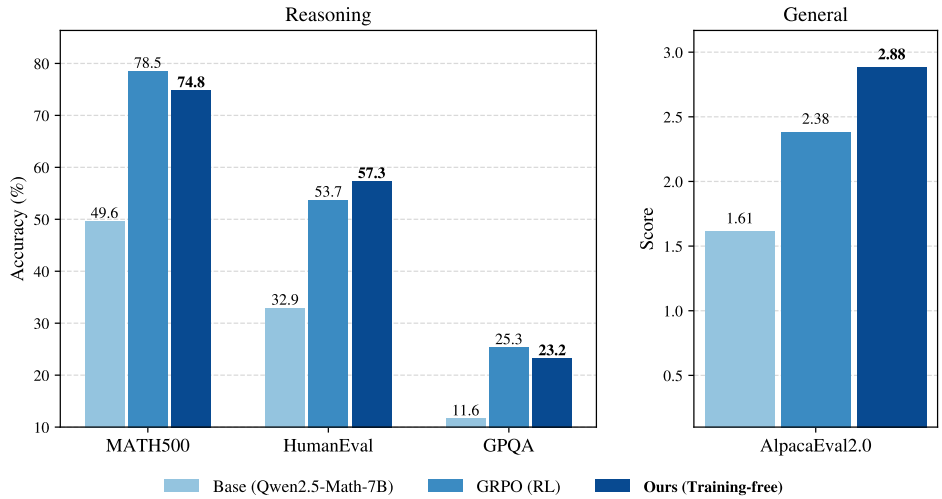

Figure 1: **Our sampling algorithm can match and outperform RL posttraining.** Left: we compare our sampling algorithm (ours) against the base model (base) and RL posttraining (GRPO) on three *verifiable reasoning* tasks (MATH500, HumanEval, GPQA). Right: we compare them on an *unverifiable general task* (AlpacaEval2.0). Our algorithm achieves comparable performance to GRPO within the posttraining domain (MATH500) but can *outperform* on out-of-domain tasks such as HumanEval and AlpacaEval.

bilities, the need to curate a diverse and expansive posttraining dataset, and the lack of guaranteed access to a ground truth verifier/reward signal Prabhudesai et al. (2025).

Our contributions can be summarized as follows:

- **i)** We introduce the *power distribution* as a useful sampling target for reasoning tasks. Since it can be explicitly specified with a base LLM, no additional training is required.
- **ii)** We further introduce an approximate sampling algorithm for the power distribution using a Markov chain Monte Carlo (MCMC) algorithm that iteratively resamples token subsequences according to their base model likelihoods.
- **iii)** We empirically demonstrate the effectiveness of our algorithm over a range of models (Qwen2.5-Math-7B, Qwen2.5-7B, Phi-3.5-mini-instruct) and reasoning tasks (MATH500, HumanEval, GPQA, AlpacaEval 2.0). Our results show that sampling directly from the base model can achieve results on par with GRPO. In fact, for some out-of-domain tasks, our algorithm consistently *outperforms* the RL baseline.

Our results collectively illustrate that existing base models are much more capable at single-shot reasoning than current sampling methods reveal.

2 Related Works

Reinforcement learning for LLMs. RL has been instrumental in posttraining LLMs. Early on, RL with human feedback (RLHF) Ouyang et al. (2022) was developed as a technique to align LLMs with human preferences using a trained reward model. Recently, RL with verifiable rewards (RLVR) has emerged as a powerful new posttraining technique, where many works Guo et al. (2025); Lambert et al. (2024); Hu et al. (2025); Zeng et al. (2025) discovered that a simple, end-of-generation reward given by an automated verifier could substantially enhance performance on difficult reasoning tasks in mathematics and coding. The Group Relative Policy Optimization (GRPO) algorithm was at the center of these advances Shao et al. (2024). Building off of this success, many subsequent works have examined using reward signals derived from internal signals such as self-entropy Zhao et al. (2025), confidence Prabhudesai et al. (2025), and even random rewards Shao et al. (2025). Similar to these works, our paper examines base model likelihoods as a mechanism for improving reasoning performance, but crucially, our technique is *training-free*.

Autoregressive MCMC sampling with LLMs. Prior works have explored integrating classic MCMC techniques with autoregressive sampling. Many settings including red-teaming, prompt-engineering, and personalized generation can be framed as targeting sampling from the base LLM distribution but *tilted* towards an external reward function. Zhao et al. (2024) proposes learning intermediate value functions that are used in a *Sequential Monte Carlo* (SMC) framework Chopin (2004), where multiple candidate sequences are maintained and updated according to their expected

future reward. Similarly, Faria et al. (2024) proposes a *Metropolis-Hastings* (MH) algorithm, which instead of maintaining multiple candidates performs iterative resampling, again updating according to expected reward. Methodologically, our sampling algorithm is most similar to this latter work, but the crucial difference is that our target sampling distribution is completely specified by the base LLM, *avoiding the need for an external reward.*

3 Preliminaries

Let $\mathcal{X}$ be a finite vocabulary of tokens, and let $\mathcal{X}^T$ denote the set of finite sequences of tokens $x_{0:T} = (x_0, x_1, \ldots, x_T)$, where $x_i \in \mathcal{X}$ for all i and $T \in \mathbb{Z}_{\geq 0}$ is some nonnegative integer. For convenience, for a given t, let $x_{<t} = (x_0, \ldots, x_{t-1})$ and $x_{>t} = (x_{t+1}, \ldots, x_T)$, with similar definitions for $x_{\leq t}$ and $x_{\geq t}$. In general, $\mathbf{x}$ refers to a token sequence $x_{0:T}$, where T is implicitly given.

Then an LLM defines a distribution p over token sequences $\mathcal{X}^T$ by autoregressively learning the conditional token distributions $p(x_t|x_{<t})$ for all t, giving the *joint distribution* via the identity

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

To sample a sequence from p, we simply sample from the LLM token by token using the conditional distributions, which by (1) directly samples from the joint distribution.

4 MCMC Sampling for Power Distributions

In this section, we introduce our sampling algorithm for base models. Our core intuition is derived from the notion of distribution sharpening posed in Section 1. *Sharpening* a reference distribution refers to reweighting the distribution so that high likelihood regions are further upweighted while low likelihood regions are downweighted, biasing samples heavily towards higher likelihoods under the reference. Then if RL posttrained models really are just sharpened versions of the base model, we should be able to explicitly specify a target sampling distribution that achieves the same effect.

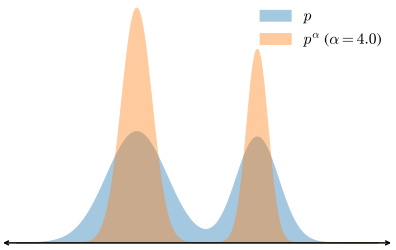

Figure 2: **A toy example of distribution sharpening.** Here p is a mixture of Gaussians, which we plot against p^α ($\alpha = 4.0$).

We organize this section as follows. Section 4.1 presents this target sharpened distribution and provides some mathematical motivation for why its samples are amenable for reasoning tasks. Section 4.2 introduces a general class of Markov chain Monte Carlo (MCMC) algorithms aimed at actually sampling from this target distribution, and finally, Section 4.3 details our specific implementation for LLMs.

4.1 Reasoning with Power Distributions

One natural way to sharpen a distribution p is to sample from the *power distribution* p^α. Since

$$p(\mathbf{x}) > p(\mathbf{x}') \implies \frac{p(\mathbf{x})^\alpha}{p(\mathbf{x}')^\alpha} > \frac{p(\mathbf{x})}{p(\mathbf{x}')} \qquad (\alpha \in [1, \infty]), \tag{2}$$

it follows that exponentiating p *increases* the relative weight on higher likelihood sequences ($\mathbf{x}$) while *decreasing* the relative weight on lower likelihood ones ($\mathbf{x}'$) (see Figure 2 for a visualization).

A related but well-known sharpening strategy is *low-temperature sampling* Wang et al. (2020), which exponentiates the conditional next-token distributions at each step:

$$p_{\text{temp}}(x_t|x_0 \ldots x_{t-1}) = \frac{p(x_t|x_{t-1} \ldots x_0)^\alpha}{\sum_{x'_t \in \mathcal{X}} p(x'_t|x_{t-1} \ldots x_0)^\alpha}, \tag{3}$$

where the *temperature* is $\tau = 1/\alpha$. A common misconception is that sampling with (3) over T tokens is equivalent to sampling from p^α; however, this is false in a subtle yet crucial way, as we illuminate in the following.

Proposition 1. *Low-temperature sampling does not sample from the power distribution p^α.*

Proof. We show that the associated conditional next-token distributions are distinct at each timestep t. The conditional distribution on x_t for p^α is given by

$$p_{\text{pow}}(x_t|x_0 \dots x_{t-1}) = \frac{\sum_{x_{>t}} p(x_0, \dots, x_t, \dots, x_T)^\alpha}{\sum_{x_{\geq t}} p(x_0, \dots, x_t, \dots, x_T)^\alpha}. \tag{4}$$

Using Bayes rule

$$p(x_t|x_{t-1} \dots x_0) = \frac{p(x_0, \dots, x_t)}{p(x_0, \dots, x_{t-1})} = \frac{\sum_{x_{>t}} p(x_0, \dots, x_t, \dots, x_T)}{\sum_{x_{\geq t}} p(x_0, \dots, x_t, \dots, x_T)}, \tag{5}$$

we can rewrite the low-temperature marginal (3) as

$$p_{\text{temp}}(x_t|x_0 \dots x_{t-1}) = \frac{\left(\sum_{x_{>t}} p(x_0, \dots, x_t, \dots, x_T)\right)^\alpha}{\sum_{x'_t} \left(\sum_{x_{>t}} p(x_0, \dots, x_t, \dots, x_T)\right)^\alpha}. \tag{6}$$

Ignoring normalizations for clarity, the relative weight on token x_t for sampling from p^α is given by a *sum of exponents*

$$p_{\text{pow}}(x_t|x_{<t}) \propto \sum_{x_{>t}} p(x_0, \dots, x_t, \dots, x_T)^\alpha. \tag{7}$$

Meanwhile, the relative weight for low-temperature sampling is given by an *exponent of sums*

$$p_{\text{temp}}(x_t|x_{<t}) \propto \left(\sum_{x_{>t}} p(x_0, \dots, x_t, \dots, x_T)\right)^\alpha. \tag{8}$$

Since the relative weights for next-token prediction for each sampling strategy are distinct, a simple expansion confirms that the distribution on sequences given by low-temperature sampling is *not the same* as the one given by p^α. □

One intuitive way to understand this difference is that low-temperature sampling does not account for how exponentiation sharpens the likelihoods of "future paths" at time step t, instead "greedily" averaging all these future likelihoods (*exponent of sums* (8)). On the other hand, sampling from p^α *inherently accounts* for future completions as it exponentiates all future paths (*sum of exponents* (7)) before computing the weights for next-token prediction. This has the following consequence:

Observation 1. *The power distribution upweights tokens with few but high likelihood future paths, while low-temperature sampling upweights tokens with several but low likelihood completions.*

Example 1. We can observe this phenomenon with a simple example. Let us consider the token vocabulary $\mathcal{X} = \{a, b\}$ and restrict our attention to two-token sequences (x_0, x_1): aa, ab, ba, bb. Let

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

so that

$$p(x_0 = a) = 0.40, \qquad p(x_0 = b) = 0.50.$$

Let $\alpha = 2.0$. Under p^α, we have

$$p_{\text{pow}}(x_0 = a) \propto 0.00^2 + 0.40^2 = 0.160, \qquad p_{\text{pow}}(x_0 = b) \propto 0.25^2 + 0.25^2 = 0.125,$$

so p^α prefers sampling a over b. Under low-temperature sampling,

$$p_{\text{temp}}(x_0 = a) \propto (0.00 + 0.40)^2 = 0.160, \qquad p_{\text{temp}}(x_0 = b) \propto (0.25 + 0.25)^2 = 0.250,$$

preferring sampling b over a. If p^α samples $x_0 = a$, there is only one future path with likelihood 0.40. If p_{temp} samples $x_0 = b$, there are two future paths ba, bb, but either choice has likelihood 0.25.

In other words, even though a has lower conditional likelihood under both p and p_{temp}, p^α upweights a and samples the highest likelihood two-token sequence. b has many future paths contributing to a higher likelihood under p and p_{temp}, but leads to low likelihood sequences. We provide a stronger formalization of this phenomenon in Appendix A.1.

Thus, sampling from p^α encourages sampling tokens which have fewer but higher likelihood "future paths", as opposed to tokens with several lower likelihood completions. This type of behavior is immensely valuable for reasoning tasks, as embedded within this target distribution is an implicit bias towards planning for high likelihood future tokens.

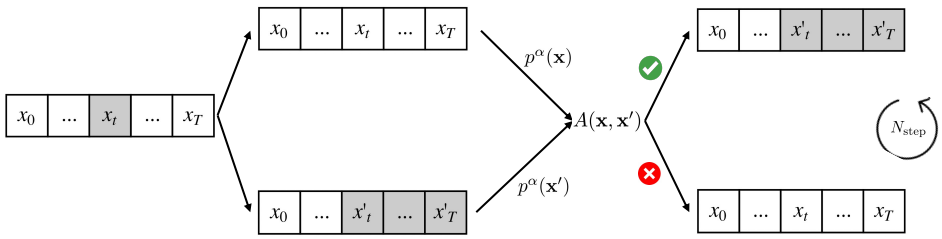

Figure 3: **Illustrating Metropolis-Hastings with random resampling.** A random index t is selected and a new candidate is generated by resampling. Based on the relative likelihoods, the candidate is accepted or rejected, and the process repeats.

4.2 The Metropolis-Hastings Algorithm

Now that we have seen how sampling from p^α can in theory assist the underlying LLM's ability to reason, our aim now turns towards proposing an algorithm to accurately sample from it. Given an LLM p, we have access to the values p^α over any sequence length; however, these values are *unnormalized*. Direct sampling from the true probabilities requires normalizing over all sequences $(x_0, \ldots, x_T) \in \mathcal{X}^T$, which is computationally intractable.

To get around this, we invoke a Markov Chain Monte Carlo (MCMC) algorithm known as Metropolis-Hastings (MH) Metropolis et al. (1953), which targets exactly what we want: approximate sampling from an unnormalized probability distribution. The MH algorithm constructs a Markov chain of sample sequences $(\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^n)$ using an arbitrary *proposal distribution* $q(\mathbf{x}|\mathbf{x}^i)$ to select the next candidate $\mathbf{x}^{i+1}$. With probability

$$A(\mathbf{x}, \mathbf{x}^i) = \min\left\{1, \frac{p^\alpha(\mathbf{x}) \cdot q(\mathbf{x}^i|\mathbf{x})}{p^\alpha(\mathbf{x}^i) \cdot q(\mathbf{x}|\mathbf{x}^i)}\right\}, \tag{9}$$

candidate $\mathbf{x}$ is accepted as $\mathbf{x}^{i+1}$; otherwise, MH sets $\mathbf{x}^{i+1} = \mathbf{x}^i$. This algorithm is especially convenient as it only requires the relative weights given by p^α (as the normalization weights in A cancel) and works with *any* generic but tractable sampler q with minimal restrictions. Remarkably, for large enough n, this process converges to sampling from the *target distribution* p^α under quite minimal conditions on the proposal distribution Neal (1993).

Consider the following family of *random resampling* proposal distributions (see Figure 3). Let p_{prop} be a proposal LLM. With uniform probability $\frac{1}{T}$, select a random $t \in [1, T]$ and *resample the sequence* starting at index t using p_{prop}. Then the transition likelihood $q(\mathbf{x}|\mathbf{x}^i)$ is simply the likelihood of the resampling. Moreover, $q(\mathbf{x}^i|\mathbf{x})$ is easy to calculate by symmetry, since we can treat $\mathbf{x}^i$ as a resampled version of $\mathbf{x}$.

With the flexibility endowed by Metropolis-Hastings, we can choose the proposal LLM p_{prop} to be any LLM with any sampling strategy (e.g., low-temperature sampling).

4.3 Power Sampling with Autoregressive MCMC

A direct implementation of Metropolis-Hastings for LLMs would involve initializing with a sampled token sequence of length T, subsequently generating new candidates of length T with (9) over many, many iterations. This process is computationally expensive, however, due to the repeated, full sequence inference calls to the LLM.

In fact, the main downside to MCMC algorithms in practice is the potential for an *exponential mixing time* Gheissari et al. (2017), where a poor choice of initialization or proposal distribution can result in an exponentially large number of samples required before convergence to the target distribution. This problem is exacerbated if the sample space has *high dimensionality* Bandeira et al. (2022); Schmidler & Woodard (2013), which is precisely exhibited by the sequence space of tokens $\mathcal{X}^T$, especially for long sequences/large values of T.

To remedy this, we propose an algorithm that leverages the sequential structure of autoregressive sampling. We define a series of intermediate distributions which we progressively sample from, until converging to the target distribution p^α. In particular, samples from one intermediate distribution initiate a Metropolis-Hastings process for the next, helping avoid pathological initializations.

Algorithm 1: Power Sampling for Autoregressive Models

Input : base p; proposal p_{prop}; power α; length T

Hyperparams: block size B; MCMC steps N_{MCMC}

Output : $(x_0, \ldots, x_T) \sim p^\alpha$

Notation: Define the unnormalized intermediate target

$$\pi_k(x_{0:kB}) \;\propto\; p(x_{0:kB})^\alpha.$$

for $k \leftarrow 0$ **to** $\lceil \frac{T}{B} \rceil - 1$ **do**

  Given prefix $x_{0:kB}$, we wish to sample from π_{k+1}. Construct initialization $\mathbf{x}^0$ by extending autoregressively with p_{prop}:

$$x_t^{(0)} \sim p_{\text{prop}}\big(x_t \mid x_{<t}\big), \qquad \text{for } kB+1 \le t \le (k+1)B.$$

  Set the current state $\mathbf{x} \leftarrow \mathbf{x}^0$.

  for $n \leftarrow 1$ **to** N_{MCMC} **do**

    Sample an index $m \in \{1, \ldots, (k+1)B\}$ uniformly.

    Construct proposal sequence $\mathbf{x}'$ with prefix $x_{0:m-1}$ and resampled completion:

$$x'_t \sim p_{\text{prop}}\big(x_t \mid x_{<t}\big), \qquad \text{for } m \le t \le (k+1)B.$$

    Compute acceptance ratio (9)

$$A(\mathbf{x}', \mathbf{x}) \;\leftarrow\; \min\left\{1,\; \frac{\pi_k(\mathbf{x}')}{\pi_k(\mathbf{x})} \cdot \frac{p_{\text{prop}}(\mathbf{x} \mid \mathbf{x}')}{p_{\text{prop}}(\mathbf{x}' \mid \mathbf{x})}\right\}.$$

    Draw $u \sim \text{Uniform}(0, 1)$;

    if $u \le A(\mathbf{x}', \mathbf{x})$ **then accept** and set $\mathbf{x} \leftarrow \mathbf{x}'$

  end

  Set $x_{0:(k+1)B} \leftarrow \mathbf{x}$ to fix the new prefix sequence for the next stage.

end

return $x_{0:T}$

Fix block size B and proposal LLM p_{prop}, and consider the sequence of (unnormalized) distributions

$$\emptyset \longrightarrow p(x_0, \ldots, x_B)^\alpha \longrightarrow p(x_0, \ldots, x_{2B})^\alpha \longrightarrow \cdots \longrightarrow p(x_0, \ldots, x_T)^\alpha, \tag{10}$$

where $p(x_0, \ldots, x_{kB})$ denotes the joint distribution over token sequences of length kB, for any k. For convenience, let π_k denote the distribution given by

$$\pi_k(x_{0:kB}) \propto p(x_{0:kB})^\alpha. \tag{11}$$

Suppose we have a sample from π_k. To obtain a sample from π_{k+1}, we initialize a Metropolis-Hastings process by sampling the next B tokens $x_{kB+1:(k+1)B}$ with p_{prop}. We subsequently run the MCMC sampling procedure for N_{MCMC} steps, using the *random resampling* proposal distribution q from the previous section. The full details are presented in Algorithm 1.

Note that Algorithm 1 is *single-shot*: even though multiple inference calls are made, the decision to accept vs. reject new tokens is made purely by base model likelihoods to simulate sampling *a single sequence* from p^α. We can interpret this as a new axis for *inference-time scaling*, as we expend additional compute during sampling to obtain a higher quality/likelihood sample.

To quantify the scaling, we can estimate the average number of tokens generated by Algorithm 1. Note that each candidate generation step when sampling from $\pi_k(x_{0:kB}$ resamples an average of $\frac{kB}{2}$ tokens, N_{MCMC} times. Summing over all k, the expected number of tokens generated is

$$\mathbb{E}_{\text{tokens}} = N_{\text{MCMC}} \sum_{k=1}^{\lceil T/B \rceil} \frac{kB}{2} \approx \frac{N_{\text{MCMC}} T^2}{4B}. \tag{12}$$

The key tradeoff here is between the block size B and number of MCMC steps N_{MCMC}. A larger B requires larger "jumps" between intermediate distributions, requiring a larger N_{MCMC} to adequately transition. In Section 5, we find an empirical value for B that makes Algorithm 1 performant for relatively small values of N_{MCMC}.

	MATH500	HumanEval	GPQA	AlpacaEval2.0
Qwen2.5-Math-7B				
Base	0.496	0.329	0.116	1.61
Low-temperature	0.690	0.512	0.207	2.09
Power Sampling (ours)	**0.748**	**0.573**	**0.232**	**2.88**
GRPO (MATH)	**0.785**	**0.537**	**0.253**	**2.38**
Qwen2.5-7B				
Base	0.498	0.329	0.217	7.05
Low-temperature	0.628	0.524	0.253	5.29
Power Sampling (ours)	**0.706**	**0.622**	**0.283**	**8.59**
GRPO (MATH)	**0.740**	**0.561**	**0.328**	**7.62**
Phi-3.5-mini-instruct				
Base	0.400	0.213	0.177	14.82
Low-temperature	0.478	0.128	0.162	**18.15**
Power Sampling (ours)	**0.508**	**0.732**	**0.263**	**17.65**
GRPO (MATH)	**0.406**	**0.134**	**0.202**	**16.74**

Table 1: **Power sampling (ours) matches and even outperforms GRPO across model families and tasks.** We benchmark the performance of our sampling algorithm on MATH500, HumanEval, GPQA, and AlpacaEval 2.0. We bold the scores of both our method and GRPO, along with the maximum score per column. Across models, we see that power sampling is comparable to GRPO on in-domain reasoning (MATH500), and can outperform GRPO on out-of-domain tasks.

5 Experiments

5.1 Experimental Setup

Evaluation. We use a standard suite of reasoning benchmarks ranging across mathematics, coding, and STEM (MATH500, HumanEval, GPQA), along with a non-verifiable benchmark (AlpacaEval 2.0) evaluating general helpfulness. We evaluate all of our methods and baselines *single-shot*; i.e., on one final response string.

- **MATH500**: The MATH dataset Lightman et al. (2024) consists of competition math problems spanning seven categories including geometry, number theory, and precalculus. There are 12500 problems total, with 7500 training problems and 5000 test problems. MATH500 is a specific randomly chosen subset of the test set standardized by OpenAI.
- **HumanEval**: HumanEval is a set of 164 handwritten programming problems covering algorihtms, reasoning, mathematics, and language comprehension Chen et al. (2021). Each problem has an average of 7.7 associated unit tests, where solving the problem corresponds to passing all unit tests.
- **GPQA**: GPQA Rein et al. (2024) is a dataset of multiple-choice science questions (physics, chemistry, and biology) which require advanced reasoning skills to solve. We use subset GPQA Diamond for evaluation, which consists of 198 questions which represent the highest quality subset of the GPQA dataset.
- **AlpacaEval 2.0**: The AlpacaEval dataset is a collection of 805 prompts Dubois et al. (2024) that gauge general helpfulness with questions asking e.g., for movie reviews, recommendations, and reading emails. The model responses are graded by an automated LLM judge (GPT-4-turbo), which determines a preference for the model responses over those from a baseline (also GPT-4-turbo). The resulting score is a win rate of model responses normalized for the length of the model response.

Models. To demonstrate the efficacy of our sampling algorithm, we use the base models Qwen2.5-Math-7B, Qwen2.5-7B, and Phi-3.5-mini-instruct. For our RL baselines, we use the implementation of GRPO in Shao et al. (2025), which posttrains these models on the MATH training split. For both the Qwen2.5 models, we use the default hyperparameters used to benchmark their performance in Shao et al. (2025). For the Phi-3.5 model, we use a set of hyperparameters selected from Abdin et al. (2024) that avoids training instabilities and converges to improvement over the base model over a large number of epochs.

Sampling Algorithm. For our implementation of power sampling (Algorithm 1), we set the maximum T to be $T_{\text{max}} = 3072$ (termination can happen earlier with an EOS token) and block size

Filter an input list of strings only for ones that start with a given prefix. (Phi-3.5-mini-instruct: HumanEval)

Method	Response	Passed
Ours	`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`

Table 2: **Sample responses on HumanEval: Phi-3.5-mini-instruct.** We present an example where our method solves a simple coding question, but GRPO does not.

$B = 16$. Empirically, we find $\alpha = 4.0$ coupled with a proposal LLM p_{prop} chosen as the base model with sampling temperature $1/\alpha$ to be most performant for reasoning tasks. For AlpacaEval 2.0, we find that having a proposal distribution of higher temperature ($\tau = 0.5$) improves performance.

5.2 Results

Main results. We display our main results in Table 1. Across base models of different families, our sampling algorithm achieves massive, near-universal boosts in single-shot accuracies and scores over different reasoning and evaluation tasks that reach, e.g., up to **+51.9%** on HumanEval with Phi-3.5-mini and **+25.2%** on MATH500 with Qwen2.5-Math. In particular, on MATH500, which is in-domain for RL posttraining, power sampling achieves accuracies that are on par with those obtained by GRPO. Furthermore, on out-of-domain reasoning, our algorithm again matches GRPO on GPQA and actually *outperforms* on HumanEval by up to **+59.8%**. Similarly, power sampling consistently outperforms on the non-verifiable AlpacaEval 2.0, suggesting a *generalizability* of our boosts to domains beyond verifiability.

The surprising success of this fundamentally simple yet training-free sampling algorithm underscores the latent reasoning capabilities of existing base models.

5.3 Analysis

Qualitative attributes of reasoning. We analyze how the reasoning characteristics of power sampling relate to those of GRPO. We present an example in Table 2, with further examples in Appendix A.2. These samples collectively illustrate that our sampling algorithm is able to find distinct reasoning traces from GRPO.

At the same time, there are some striking similarities. By design, power sampling targets sampling higher likelihood sequences from the base model. In Figure 4, the left graph plots a histogram of the log likelihoods (normalized by length) of base model, power sampling, and GRPO responses on MATH500, where likelihoods are taken relative to the Qwen2.5-Math-7B base model. Our method samples from higher likelihood regions of the base model, as intended, but still maintains noticeable spread. Meanwhile, GRPO samples are heavily concentrated at the highest likelihood peak. This aligns with the observation that RL posttraining strongly sharpens the base model distribution at the expense of diversity Song et al. (2025). While Section 4.1 only heuristically relates better reasoning performance to higher likelihood sampling, Figure 4 illustrates empirically that correct reasoning traces fall in higher likelihood regions of the base model distribution.

We also plot the base model *confidence* of MATH500 responses, defined to be the negative entropy (*uncertainty*) of the token sequence. The right plot of Figure 4 demonstrates that our method's and GRPO responses sample from similarly high confidence regions from the base model, which again correspond to regions of higher likelihood and correct reasoning.

Finally, another defining characteristic of RL-posttraining is long-form reasoning Guo et al. (2025), where samples tend to exhibit longer responses. On MATH500, Qwen2.5-Math-7B averages a response length of **600** tokens, while GRPO averages **671** tokens. Surprisingly, power sampling achieves a similar average length of **679** tokens, *without explicitly being encouraged to favor longer generations.* This emerges naturally from the sampling procedure.

The effect of power distributions. The two most important hyperparameters for power sampling are the choice of α and the number of MCMC (resampling) steps during sequence generation N_{MCMC}. At the extremes, choosing $\alpha = 1.0$ samples from the base model directly, while taking $\alpha \to \infty$ has the effect of deterministically accepting any resampled sequence that strictly increases the likelihood. Of course, even though higher base model likelihoods correlate with better reasoning

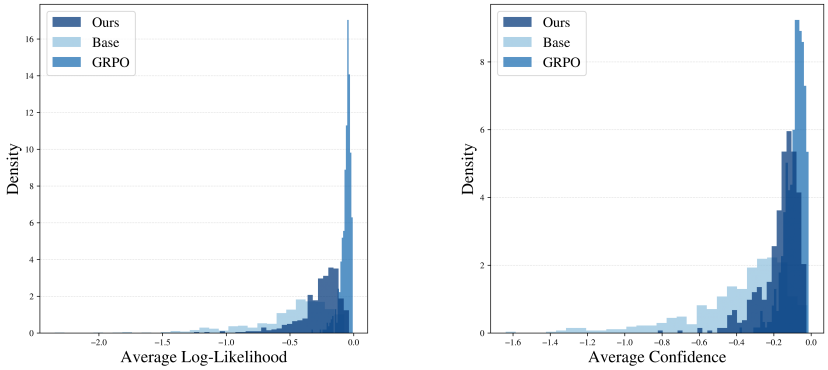

Figure 4: **Base model (Qwen2.5-Math-7B) likelihoods and confidences for MATH500 responses.** Left: We plot the log likelihoods (relative to the base model) of original, power sampling, and GRPO responses over MATH500. Right: We do the same but for confidences relative to the base model. We observe that GRPO samples from the highest likelihood and confidence regions with power sampling close behind, which correlates with higher empirical accuracy.

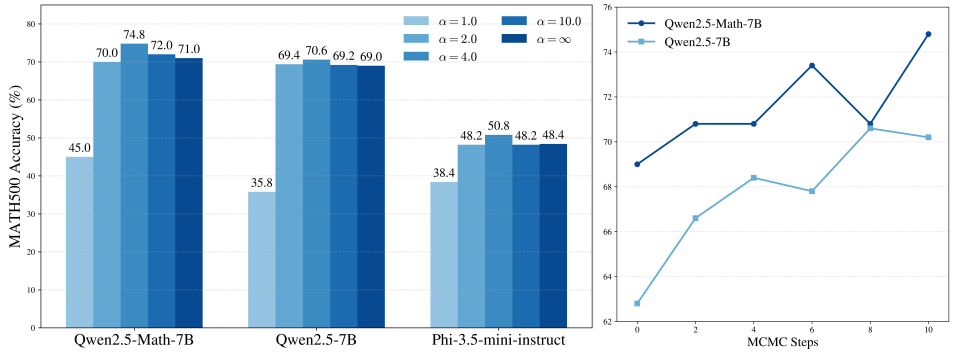

Figure 5: **Effect of hyperparameters on power sampling.** Left: We plot MATH500 accuracy across model families for various values of α. Right: We plot the increase in accuracy of power sampling on Qwen models as the number of MCMC steps increases.

(Figure 4), directly optimizing for likelihood is not necessarily optimal for reasoning, suggesting an ideal intermediate value of α.

In Figure 5, we display MATH500 accuracies across various values of α and find that an intermediate $\alpha = 4.0$ outperforms other values, as expected. Noticeably, the accuracies of power sampling remain relatively stable beyond $\alpha \geq 2.0$, suggesting that power sampling in practice is relatively robust to the choice of α.

Test-time scaling with MCMC steps. On the other hand, N_{MCMC} toggles the inference-time compute expended by our algorithm, providing a natural axis for test-time scaling. In Section 4.3 we raised the notion of a *mixing time*, or the number of MCMC steps required before adequately sampling from the target distribution. In our case, we expect that the fewer MCMC steps we take, the further our algorithm samples from the target p^α.

We plot performance dependence on N_{MCMC} in Figure 5 and notice a steady increase in accuracy until $N_{\text{MCMC}} = 10$, beyond which accuracy stabilizes, as expected. The accuracy difference from using fewer MCMC steps is noticeable but no more than 3-4% between $N_{\text{MCMC}} = 2$ and $N_{\text{MCMC}} = 10$. However, the jump in accuracy by using at least two steps as opposed to none is substantial (3-4%).

6 Conclusion

In this work, we present an algorithm that samples directly from a base model without any additional training or access to an external signal, achieving a single-shot reasoning performance that is on par with, and sometimes even better than, that of a state-of-the-art RL posttraining algorithm. Our results suggest that base model capabilities are underutilized at sampling time and point towards a close relationship between high likelihood regions of the base model and strong reasoning capabilities. Employing additional compute at sampling-time with a stronger understanding of base model capabilites offers a promising direction for expanding the scope of reasoning beyond verifiability.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J. Hewett, Mojan Javaheripi, Piero Kauffmann, James R. Lee, Yin Tat Lee, Yuanzhi Li, Weishung Liu, Caio C. T. Mendes, Anh Nguyen, Eric Price, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Xin Wang, Rachel Ward, Yue Wu, Dingli Yu, Cyril Zhang, and Yi Zhang. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024. URL https://arxiv.org/abs/2412.08905. 7

Afonso S. Bandeira, Antoine Maillard, Richard Nickl, and Sven Wang. On free energy barriers in gaussian priors and failure of cold start mcmc for high-dimensional unimodal distributions. *arXiv preprint arXiv:2209.02001*, 2022. URL https://arxiv.org/abs/2209.02001. 5

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374. 7

Nicolas Chopin. Central limit theorem for sequential monte carlo methods and its application to bayesian inference. *The Annals of Statistics*, 32(6):2385–2411, 2004. doi: 10.1214/009053604000000615. URL https://projecteuclid.org/journals/annals-of-statistics/volume-32/issue-6/Central-limit-theorem-for-sequential-Monte-Carlo-methods-and-its/10.1214/009053604000000698.full. 2

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024. URL https://arxiv.org/abs/2404.04475. 7

Gonçalo R. A. Faria, Sweta Agrawal, António Farinhas, Ricardo Rei, José G. C. de Souza, and André F. T. Martins. Quest: Quality-aware metropolis-hastings sampling for machine translation. In *NeurIPS*, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/a221d22ff6a33599142c8299c7ed06bb-Paper-Conference.pdf. 3

Reza Gheissari, Eyal Lubetzky, and Yuval Peres. Exponentially slow mixing in the mean-field swendsen–wang dynamics. *arXiv preprint arXiv:1702.05797*, 2017. 5

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 1, 2, 8

Andre He, Daniel Fried, and Sean Welleck. Rewarding the unlikely: Lifting GRPO beyond distribution sharpening. *arXiv preprint arXiv:2506.02355*, 2025. 1

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. *arXiv preprint arXiv:2103.03874*, 2021. 1

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. *arXiv preprint arXiv:2503.24290*, 2025. 1, 2

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024. 2

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 Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with AlphaCode. *arXiv preprint arXiv:2203.07814*, 2022. 1

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. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi. 7

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. doi: 10.1063/1.1699114. 5

Radford M Neal. Probabilistic inference using markov chain monte carlo methods. *Department of Computer Science, University of Toronto (review paper / technical report)*, 1993. 5

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *NeurIPS*, volume 35, pp. 27730–27744, 2022. 2

Mihir Prabhudesai, Lili Chen, Alex Ippoliti, Katerina Fragkiadaki, Hao Liu, and Deepak Pathak. Maximizing confidence alone improves reasoning. *arXiv preprint arXiv:2505.22660*, 2025. URL https://arxiv.org/abs/2505.22660. 2

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 q&a benchmark. In *First Conference on Language Modeling*, 2024. 1, 7

Scott C. Schmidler and Dawn B. Woodard. Lower bounds on the convergence rates of adaptive mcmc methods. Technical report, Duke University / Cornell University, 2013. URL https://www2.stat.duke.edu/~scs/Papers/AdaptiveLowerBounds_AS.pdf. 5

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. *arXiv preprint arXiv:2506.10947*, 2025. 1, 2, 7

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseek-math: Advancing mathematical reasoning through step-by-step exploration. *arXiv preprint arXiv:2404.01140*, 2024. 1, 2

Yuda Song, Julia Kempe, and Rémi Munos. Outcome-based exploration for LLM reasoning. *arXiv preprint arXiv:2509.06941*, 2025. URL https://arxiv.org/abs/2509.06941. 1, 8

Pei-Hsin Wang, Sheng-Iou Hsieh, Shih-Chieh Chang, Yu-Ting Chen, Jia-Yu Pan, Wei Wei, and Da-Chang Juan. Contextual temperature for language modeling. *arXiv preprint arXiv:2012.13575*, 2020. URL https://arxiv.org/abs/2012.13575. 3

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? *arXiv preprint arXiv:2504.13837*, 2025. URL https://arxiv.org/abs/2504.13837. 1

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerlzoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025. URL https://arxiv.org/abs/2503.18892. 2

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Grosse. Probabilistic inference in language models via twisted sequential monte carlo. *arXiv preprint arXiv:2404.17546*, 2024. URL https://arxiv.org/abs/2404.17546. 2

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. *arXiv preprint arXiv:2505.19590*, 2025. URL https://arxiv.org/abs/2505.19590. 2

A Appendix

A.1 Additional Theoretical discussion

Proposition 2. *Power sampling upweights tokens with small support but high likelihood completions, while low-temperature sampling upweights tokens with large support but low likelihood completions.*

Proof. Suppose we have sampled $x_{0:t-1}$, and consider two choices of next-token x_t and x'_t. Suppose that the likelihood of sampling $(x_0, \ldots, x_t)$ under p is ε but has singular support; i.e., for only one choice of $x_{>t}$ is $p(x_0, \ldots, x_t, \ldots, x_T)$ nonzero, so that $p(x_0, \ldots, x_t, \ldots, x_T) = \varepsilon$. On the other hand, suppose the likelihood of $(x_0, \ldots, x'_t)$ under p has mass ε' uniformly distributed across a support of size N; i.e., there exist N completions $x_{>t}$ such that $p(x_0, \ldots, x'_t, \ldots, x_T)$ is nonzero and has mass $\frac{\varepsilon'}{N}$.

If $\varepsilon < \varepsilon'$, then under the naïve sampler, the relative marginal weights on x_t and x'_t are ε^α and ε'^α, so the probability of choosing x_t is *downweighted* relative to x'_t. However, for the power distribution, the relative marginal weights are $p_{\text{pow}}(x_t|x_{<t}) = \varepsilon^\alpha$ and $p_{\text{pow}}(x'_t|x_{<t}) = \frac{\varepsilon'^\alpha}{N^{\alpha-1}}$. Then, as long as $\varepsilon^\alpha > \frac{\varepsilon'^\alpha}{N^{\alpha-1}} \iff \varepsilon > \frac{\varepsilon'}{N^{1-1/\alpha}}$, token x_t will be *upweighted* relative to token x'_t. In particular, since $\alpha \geq 1$, we have the inequalities $\frac{\varepsilon'}{N} < \frac{\varepsilon'}{N^{1-1/\alpha}} < \varepsilon < \varepsilon'$.

In other words, the likelihood mass on $(x_0, \ldots, x_t)$ can be *less than* the mass on $(x_0, \ldots, x'_t)$ under p, but the sparse support implies that the completion for x_t is of higher likelihood under than any individual completion for x'_t, so the power distribution favors sampling x_t over x'_t. □

A.2 More Qualitative Examples

Can you solve the following math problem? Write $\frac{3}{20}$ as a decimal. Please reason step by step, and put your final answer within \boxed{{}}. Correct answer: $\boxed{0.15}$

System	Answer
Ours	To convert the fraction $\frac{3}{20}$ to a decimal, we can follow these steps: 1. Identify the numerator and the denominator of the fraction. In this case, the numerator is 3 and the denominator is 20. 2. Divide the numerator by the denominator. In this case, we divide 3 by 20. 3. Perform the division. The result is 0.15. Therefore, the fraction $\frac{3}{20}$ as a decimal is $\boxed{0.15}$.
GRPO	To convert the fraction $\frac{3}{20}$ to a decimal, we can follow these steps: 1. **Understand the fraction**: The fraction $\frac{3}{20}$ means 3 divided by 20. 2. **Perform the division**: We need to divide 3 by 20. Since 3 is less than 20, we will add a decimal point and zeros to 3 to make the division possible. 3. **Set up the division**: Write 3 as 3.000 (adding three zeros for precision). 4. **Divide**: - 20 goes into 30 once (since $20 \times 1 = 20$). - Subtract 20 from 30 to get a remainder of 10. - Bring down the next 0 to make it 100. - 20 goes into 100 five times (since $20 \times 5 = 100$). - Subtract 100 from 100 to get a remainder of 0. 5. **Combine the results**: The quotient from the division is 0.15. Therefore, the decimal representation of $\frac{3}{20}$ is $\boxed{0.15}$.

Table 3: Comparison on MATH500: Qwen2.5-Math-7B. We present an example where our method and GRPO are correct.

The Fib4 number sequence is a sequence similar to the Fibonacci sequence defined as: fib4(0)=0, fib4(1)=0, fib4(2)=2, fib4(3)=0, and for $n \geq 4$, fib4(n) = fib4(n-1) + fib4(n-2) + fib4(n-3) + fib4(n-4). Write a function to efficiently compute the n-th element without recursion.

System	Completion	Passed

Ours	`# Base cases`
	`if n == 0:`
	`  return 0`
	`elif n == 1:`
	`  return 0`
	`elif n == 2:`
	`  return 2`
	`elif n == 3:`
	`  return 0`
	
	`# Initialize the first four elements`
	`fib4_values = [0, 0, 2, 0]`
	
	`# Compute the rest of the sequence iteratively`
	`for i in range(4, n+1):`
	`  next_value = (fib4_values[i-1] + fib4_values[i-2] +`
	`  fib4_values[i-3] + fib4_values[i-4])`
	`  fib4_values.append(next_value)`
	
	`# Return the n-th element`
	`return fib4_values[n]`
		`true`

GRPO	`a, b, c, d = 0, 0, 2, 0`
	`for _ in range(n):`
	`  a, b, c, d = b, c, d, a + b + c + d`
	`return d`
		`false`

Table 4: HumanEval comparison between our .

Can you solve the following math problem? What integer n satisfies $0 \leq n < 18$ and

$$n \equiv -11213141 \pmod{18}\ ?$$

Please reason step by step, and put your final answer within \boxed{{}}. Correct answer: $\boxed{13}$

Method	Answer
Ours	First, we perform the division of 11213141 by 18: $$11213141 \div 18 \approx 622952.277777\ldots$$ The integer part of the quotient is 622952. Now, we multiply 622952 by 18 to find the product: $$622952 \times 18 = 11213136$$ Next, we subtract this product from 11213141 to find the remainder: $$11213141 - 11213136 = 5$$ So, the remainder when 11213141 is divided by 18 is 5. This means: $$11213141 \equiv 5 \pmod{18}$$ Therefore: $$-11213141 \equiv -5 \pmod{18}$$ Since we need a non-negative remainder, we convert -5 to a positive equivalent by adding 18: $$-5 + 18 = 13$$ Thus: $$-11213141 \equiv 13 \pmod{18}$$ The integer n that satisfies $0 \leq n < 18$ and $n \equiv -11213141 \pmod{18}$ is: $\boxed{13}$.
GRPO	Step 1: Find the remainder when 11213141 is divided by 18. We can use the property of modular arithmetic that states $a \equiv b \pmod{m}$ if and only if $a \mod m = b \mod m$. So, we need to compute $11213141 \mod 18$. To simplify this, we can use the fact that the remainder of a number when divided by 18 is the same as the remainder of the sum of its digits when divided by 18 (although this is not always the most efficient method, it can be used here for simplicity). First, let's sum the digits of 11213141: $$1 + 1 + 2 + 1 + 3 + 1 + 4 + 1 = 14.$$ Now, we need to find the remainder when 14 is divided by 18. Since 14 is less than 18, the remainder is simply 14. Therefore, $$11213141 \equiv 14 \pmod{18}.$$ Step 2: Find the remainder when -11213141 is divided by 18. Since $11213141 \equiv 14 \pmod{18}$, it follows that $$-11213141 \equiv -14 \pmod{18}.$$ We need to convert -14 to a positive remainder between 0 and 17. To do this, we add 18 to -14: $$-14 + 18 = 4.$$ Therefore, $$-11213141 \equiv 4 \pmod{18}.$$ The integer n that satisfies $0 \leq n < 18$ and $n \equiv -11213141 \pmod{18}$ is $\boxed{4}$.

Table 5: MATH500 comparison between our sampling algorithm and GRPO for Qwen2.5-Math-7B. Here is an example where GRPO gets an incorrect answer, while our sampling algorithm succeeds. Our sample answer uses a distinct method altogether.