A CLOSER LOOK AT POWER SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in the reasoning capabilities of language models have been driven largely by the use of reinforcement learning with verifiable rewards (RLVR) in post-training pipelines. However, recent work has suggested that, at least with an open-source compute budget, RLVR may improve language-model performance primarily by increasing the probability of sequences that already have high probability under the base model. This has motivated the creation of training-free, test-time methods for sharpening the base model distribution based on the *power distribution*: a global tempering of the language model distribution that up-weights high-likelihood sequences. Prior work has shown that sampling approximately from the power distribution leads to strong accuracy for certain models and benchmarks. In this paper, we create a novel exact sampler for the power distribution and devise experiments for probing the power distribution. Our experiments show that exact samples from the power distribution closely resemble samples generated with likelihood-maximization-based beam search. The power distribution improves accuracy for some models and benchmarks and drastically degrades accuracy on others, challenging the notion that that performance gains observed with RLVR can be attributed to its use. Therefore, past gains observed from approximate sampling from the power distribution must be attributed to their sampling strategy rather than the underlying target distribution.

1 INTRODUCTION

Early progress in language-model reasoning was driven by large-scale next-token prediction pre-training (Kaplan et al., 2020; Brown et al., 2020). More recent advances have come from post-training with reinforcement learning with verifiable rewards (RLVR) (Shao et al., 2024; DeepSeek, 2025; OpenAI, 2024). Post-training is costly: it requires extensive rollouts and often relies on data that are difficult to collect, such as human judgments or synthetic data that are not publicly available. A common motivation for this cost is that RLVR enables models to acquire capabilities absent from the base model.

A competing hypothesis holds that, at least at the compute budgets available to open-source projects (Shao et al., 2025), these gains need not involve new capabilities. Under this view, post-training raises benchmark accuracy primarily by concentrating probability mass on completions to which the base model already assigns relatively high probability (Yue et al., 2025; Shao et al., 2025; He et al., 2025; Huang et al., 2025). This is known as the *distribution sharpening* hypothesis.

The hypothesis becomes actionable once the sharpened distribution is given an explicit form. Karan & Du (2026) propose the *power distribution*, a global tempering of the base model that raises the probability of each completion to a power $\alpha > 1$ and renormalizes over complete sequences. Because this distribution is defined entirely in terms of the base model, it can be targeted at test-time by approximate inference, without rewards or further training. Any algorithm that samples approximately from the power distribution with test-time compute is known as *power sampling*. If exact power sampling captures the effect of RLVR, part of the cost of post-training could be replaced by test-time computation, and approximate inference would constitute an additional axis for scaling capabilities (Karan & Du, 2026). As such, prior work has sought more computationally efficient algorithms for converging to the power distribution, from Metropolis–Hastings (Karan & Du, 2026) to sequential Monte Carlo and related methods (Ji et al., 2026; Azizi et al., 2026; Zhou et al., 2026; Meng et al., 2026), and has reported improved benchmark accuracy.

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.

Yet, these results leave open whether the improvements come from the power distribution itself or from the approximation method used to reach it. We separate the two with an exact sampler for the power distribution. Across four models and five mathematics and code benchmarks, we find that the power distribution at $\alpha = 4$ behaves as an approximation to its $\alpha = \infty$ limit, which places all of its mass on the completions with the highest likelihood. Its accuracy closely tracks that of beam search, so it rises or falls with the quality of the most probable completion. The power distribution therefore reproduces the behavior of likelihood-maximization decoding rather than that of RLVR.

We make the following contributions:

An exact sampler for the power distribution. We present a rejection sampler that produces exact samples from the power distribution when certain conditions are met (Section 3.1). To increase the number of problems on which it accepts a sample, we use a highly vectorized GPU implementation. It yields samples on over 57% of the problems we evaluate. Because these samples come from the power distribution itself, they isolate its properties from artifacts of test-time approximate inference and of training dynamics involved in fine-tuning a policy toward it (Mittal et al., 2026).

Exact power sampling resembles beam search rather than RLVR. Across four models and five benchmarks, accuracy under the power distribution at $\alpha = 4$ closely tracks that of best-first beam search (Table 1). Importance-sampling estimates show that this behavior emerges at small values of α (Figure 1), and completions shorten to lengths similar to those of beam search (Figure 2). The power distribution therefore inherits the accuracy of the highest-likelihood completion, improving on autoregressive sampling in some settings and degrading it in others, whereas RLVR tends to improve accuracy and lengthen completions (Shao et al., 2024) in general.

2 PRELIMINARIES

Completions and the language-model distribution. Let Σ be a token vocabulary containing an end-of-sequence token `EOS`. We write Σ^* for the set of finite completions $\boldsymbol{y} = (y_1, \ldots, y_t)$ that end in `EOS` and contain no earlier `EOS` token. A prompt $\boldsymbol{x}$ is fixed conditioning context and is not part of the completion. The base language model defines next-token probabilities and hence the completion probability

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{i=1}^{t} p(y_i \mid \boldsymbol{x}, y_{<i}), \tag{1}$$

where $y_{<i} = (y_1, \ldots, y_{i-1})$. The product includes the probability of `EOS`. We assume termination with probability one, so that $\sum_{\boldsymbol{y} \in \Sigma^*} p(\boldsymbol{y} \mid \boldsymbol{x}) = 1$. Autoregressive sampling draws each token from these conditional probabilities at temperature 1 until `EOS` is emitted.

The power distribution. For $\alpha \geq 1$, the power distribution over completions is

$$p_\alpha(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{x})^\alpha}{Z_\alpha(\boldsymbol{x})}, \qquad Z_\alpha(\boldsymbol{x}) = \sum_{\boldsymbol{y}' \in \Sigma^*} p(\boldsymbol{y}' \mid \boldsymbol{x})^\alpha. \tag{2}$$

At $\alpha = 1$, this is the base-model distribution. For $\alpha > 1$, it increases the relative probability of higher-probability completions. This normalization is over complete outputs; it generally differs from applying a temperature of $1/\alpha$ separately at each token.

The argmax completion. An argmax completion is any globally most probable completion,

$$\boldsymbol{y}^\star \in \arg\max_{\boldsymbol{y} \in \Sigma^*} p(\boldsymbol{y} \mid \boldsymbol{x}). \tag{3}$$

Thus $p(\boldsymbol{y}^\star \mid \boldsymbol{x})$ bounds the probability of every completion for the prompt. The same completions maximize p_α for every $\alpha \geq 1$. The limit of the power distribution $p_\alpha(\boldsymbol{y} \mid \boldsymbol{x})$ converges to a distribution that assigns 0 to non-argmax completions and uniformly weights the argmax completions.

Beam search. Because Σ^* is infinite, exact computation of Equation 3 is generally intractable, and beam search is the standard heuristic for approximating it. In our experiments, we use a variant of beam search based on best-first beam search (Meister et al., 2020). The algorithm is simple: it iteratively pops the the highest-likelihood sequence from the priority queue, creates a list of candidates by concatenating the sequence with all of the tokens that can be sampled next, then inserts these candidates in the priority queue ranked by likelihood. The algorithm proceeds in this manner until either a timeout is reached or an `EOS`-terminated sequence is popped from the queue. If an `EOS`-terminated sequence has a higher probability than the sequence at the top of the queue, then it must be the highest-likelihood sequence in the language model's probability distribution—all other sequences in the language model distribution are seeded by some sequence that was in the priority queue when the `EOS`-terminated sequence was popped and concatenating a token to a sequence can only lower its likelihood. Though finding the highest-likelihood completion is not guaranteed, the completions obtained by beam search typically have much higher likelihood than those produced by autoregressive decoding or temperature sampling.

3 METHODS

We study the power distribution with three inference procedures, each with a different role, and compare them with autoregressive sampling from the base model at temperature 1 and beam search sampling. Rejection sampling (Section 3.1) produces exact samples from the power distribution on problems where an argmax completion can be certified and at least one proposal is accepted; it is the reference against which we judge the other procedures. As rejection-sampling estimates are only available on a subset of benchmark tasks, estimates reported below are averaged over problems where at least one rejection sample was available. Self-normalized importance sampling (Section 3.2) estimates expectations under the power distribution on every problem, including problems where rejection sampling produces no sample. To understand the sensitivity of the Metropolis–Hastings-based inference algorithm offered by Karan & Du (2026) to its initial starting point, we run two instantiations of their algorithm. First, we run their original Metropolis–Hastings algorithm. Then, we run the same algorithm but we initialize it at a completion obtained with beam search.

3.1 EXACT SAMPLING WITH REJECTION SAMPLING

Rejection sampling produces exact samples from $p_\alpha(\cdot \mid \boldsymbol{x})$ in a two-stage procedure (Algorithm 1). First, it searches for an argmax completion $\boldsymbol{y}^\star$ (Equation 3) together with a certificate that no completion is more probable. Second, it draws proposals from the base model and accepts each proposal $\boldsymbol{y}$ with probability $\big(p(\boldsymbol{y} \mid \boldsymbol{x})/p(\boldsymbol{y}^\star \mid \boldsymbol{x})\big)^{\alpha-1}$. Accepted proposals are exact samples from p_α; Appendix B proves the correctness of rejection sampling.

Rejection sampling returns samples for a problem only when the search certifies an argmax and at least one proposal is accepted. To make both events more likely, we implement both stages as batched GPU computations. The search is a best-first search over prefixes (Meister et al., 2020) that uses batch inference to search different parts of the distribution in parallel. Rather than popping just a single sequence from the priority queue as described above, we pop the top-256 sequences in parallel and compute the probabilities of their next tokens. At the proposal stage, rather than sampling a single sequence at a time and checking if it is accepted, we generate size-1024 batches and return all completions that are accepted.

3.2 APPROXIMATION WITH SELF-NORMALIZED IMPORTANCE SAMPLING

Rather than producing samples, self-normalized importance sampling estimates expectations under the power distribution. For a prompt $\boldsymbol{x}$ and a statistic $f(\boldsymbol{x}, \boldsymbol{y})$, we draw N completions $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N$ independently from the base model at temperature 1 and estimate $\mathbb{E}_{\boldsymbol{Y} \sim p_\alpha(\cdot \mid \boldsymbol{x})}[f(\boldsymbol{x}, \boldsymbol{Y})]$ by

$$\hat{\mathbb{E}}_\alpha^N\big[f \mid \boldsymbol{x}\big] = \sum_{i=1}^{N} \bar{w}_i \, f(\boldsymbol{x}, \boldsymbol{y}_i), \qquad \bar{w}_i = \frac{p(\boldsymbol{y}_i \mid \boldsymbol{x})^{\alpha-1}}{\sum_{\ell=1}^{N} p(\boldsymbol{y}_\ell \mid \boldsymbol{x})^{\alpha-1}}. \tag{4}$$

The weight $p(\boldsymbol{y} \mid \boldsymbol{x})^{\alpha-1}$ is the ratio of the unnormalized target $p(\boldsymbol{y} \mid \boldsymbol{x})^\alpha$ to the proposal $p(\boldsymbol{y} \mid \boldsymbol{x})$. Normalizing the weights within each prompt removes the need for $Z_\alpha(\boldsymbol{x})$. This is a biased but

Algorithm 1 Exact rejection sampling from the power distribution.

```
def rejection_sample(model, x, alpha, L, k=256, b=128, tau=180, B=1024):
    """Exact samples from p_alpha(. | x), or [] if no argmax is certified."""
    # Stage 1: certify an argmax completion with best-first search.
    y_star = greedy_decode(model, x, max_len=L)
    queue = PriorityQueue()              # prefixes, most probable first
    queue.push(EMPTY_PREFIX, logp=0.0)
    certified, start = True, time.time()
    while queue and time.time() - start < tau:
        for y in queue.pop_best(b):      # b most probable prefixes, one batch
            if model.logp(x, y) <= model.logp(x, y_star):
                continue
            if y.rank == k:              # rank of y's last token among parent's top-k
                certified = False        # omitted tokens could lead higher
            if y.is_complete(max_len=L):
                y_star = y
            else:
                for a in model.top_k_next_tokens(x, y, k):
                    queue.push(y + a, logp=model.logp(x, y + a))
        if model.logp(x, y_star) >= queue.max_logp():
            break                        # nothing left in Q can beat y_star
    if time.time() - start >= tau:
        certified = False
    if not certified:
        return []

    # Stage 2: accept or reject proposals from the base model.
    ys = model.sample(x, n=B, temperature=1.0, max_len=L)   # in parallel
    log_ratio = (alpha - 1) * (model.logp(x, ys) - model.logp(x, y_star))
    log_u = np.log(np.random.rand(B))
    return [y for y, lu, lr in zip(ys, log_u, log_ratio) if lu <= lr]
```

consistent estimator of $\mathbb{E}_{\boldsymbol{Y}\sim p_\alpha(\cdot|\boldsymbol{x})}[f(\boldsymbol{x},\boldsymbol{Y})]$. To estimate accuracy on a benchmark of prompts $\{\boldsymbol{x}_m\}_{m=1}^M$, we set $f(\boldsymbol{x},\boldsymbol{y}) = 1$ if $\boldsymbol{y}$ is a correct solution to $\boldsymbol{x}$ and 0 otherwise, and average over prompts:

$$\hat{\mu}_\alpha = \frac{1}{M}\sum_{m=1}^{M} \hat{\mathbb{E}}_\alpha^N\big[f \mid \boldsymbol{x}_m\big]. \tag{5}$$

Other statistics, such as the histogram of completion lengths in Figure 2 and the fraction of completions of at most five tokens in Figure 3, are estimated in a similar way. We produce $N = 1{,}024$ proposals per prompt using batch inference.

3.3 Metropolis–Hastings with Beam-search Initialization

We use the Metropolis–Hastings sampler from Karan & Du (2026) in our experiments. It builds a completion in blocks of S tokens. At stage j, it targets the power distribution restricted to completions of at most $L_j = jS$ tokens, extends the current completion to the new length cap, and performs a fixed number of Metropolis–Hastings updates. Each update chooses a position in the current completion uniformly at random, keeps the prefix before it, resamples the suffix from the base model at temperature $1/\alpha$, and accepts the result with the Metropolis–Hastings acceptance probability. We use blocks of $S = 192$ tokens and 10 updates per block.

We run this sampler from two initializations. The first sampler starts from an empty completion. It draws every proposed suffix at temperature $1/\alpha$, which concentrates on tokens that are probable at each step and rarely produces the improbable first tokens with which degenerate completions such as "ôt" begin (Figure 4). Because such completions can carry much of the mass of the power distribution (Section 4.2), a chain that rarely proposes them may not reach the power distribution within a practical number of updates. The second initialization uses the same proposal distribution and incremental staging scheme, but is initialized from the output of best-first beam search. Starting

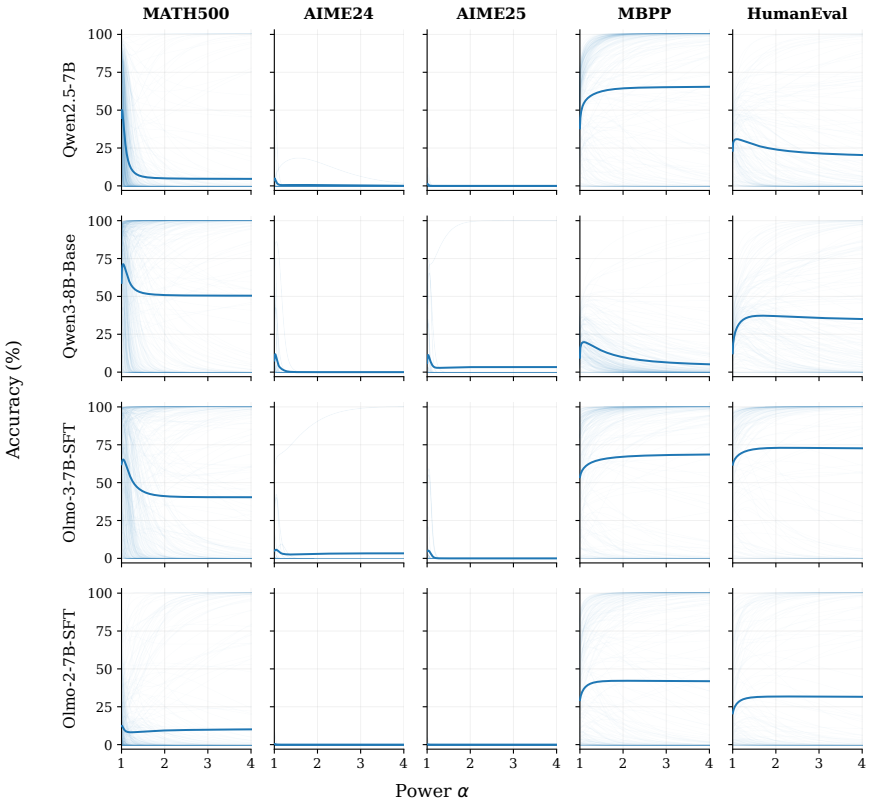


Figure 1: **Estimated accuracy (%) under the power distribution as a function of** α. These accuracy estimates use self-normalized importance sampling with 1,024 temperature-1 completions per problem (Equation 4). Rows correspond to models and columns to benchmarks. Faint curves show individual problems; bold curves show benchmark averages, with weights normalized within each problem.

from the beam-search output tests the opposite direction. Nearly every update from this state proposes a different, typically longer completion, so if the power distribution placed little mass near the beam-search output, the chain would have many opportunities to leave it. A chain that stays near its starting point despite these proposals therefore suggests that the power distribution concentrates there.

4 Experiments

4.1 Experimental setup

We evaluate four language models—Qwen2.5-7B, Qwen3-8B-Base, Olmo-2-7B-SFT, and Olmo-3-7B-SFT—on mathematics (MATH500, AIME24, AIME25) and code generation (MBPP, HumanEval). The Qwen models connect our evaluation to prior work on sampling from the power distribution (Karan & Du, 2026). We select Olmo for its openly documented training data and procedures. We use its SFT checkpoints for their standardized chat templates.

We compare six methods. Two do not target the power distribution: autoregressive sampling from the full vocabulary at temperature 1, and beam search. The other four target the power distribution with $\alpha = 4$: rejection sampling, importance sampling, and Metropolis–Hastings from the empty completion and from the beam-search output. Both Metropolis–Hastings samplers perform 10 updates per 192-token block. All methods are evaluated on the full benchmarks except rejection sampling, which we evaluate only on problems with a certified argmax and at least one accepted proposal; we report the number of such problems with each result. Appendix H gives further details.

Table 1: **Model accuracy (%) under different sampling algorithms.** The methods are autoregressive sampling (AR), Metropolis–Hastings from the empty completion (MH), beam search (BS), rejection sampling (RS), self-normalized importance sampling (SNIS), and Metropolis–Hastings from the beam-search output (MH-BS). RS, SNIS, and MH-BS estimate accuracy under the power distribution; MH shows the result of the prior-work initialization. RS includes only problems with a saved argmax certificate and at least one accepted proposal, and its brackets give qualifying/full benchmark problem counts. RS therefore evaluates a different subset from the other methods.

Model	Method	MATH500	AIME24	AIME25	MBPP	HumanEval
Qwen2.5-7B	AR	44.79	3.68	1.00	38.02	23.21
	MH	66.60	16.67	6.67	71.21	52.44
	BS	5.20	0.00	0.00	63.81	20.73
	RS	3.94 [165/500]	0.00 [10/30]	0.00 [11/30]	69.16 [231/257]	20.83 [163/164]
	SNIS	4.64	0.07	0.00	65.42	20.37
	MH-BS	5.40	0.00	0.00	64.20	20.12
Qwen3-8B-Base	AR	58.92	7.64	5.96	9.52	12.40
	MH	78.00	16.67	16.67	24.12	59.76
	BS	68.80	10.00	10.00	3.11	32.93
	RS	77.46 [71/500]	0.00 [1/30]	0.00 [1/30]	5.50 [252/257]	35.54 [161/164]
	SNIS	50.47	0.00	3.33	5.17	35.06
	MH-BS	67.00	13.33	10.00	5.06	34.76
Olmo-3-7B-SFT	AR	62.11	5.09	4.63	53.63	61.86
	MH	73.60	10.00	13.33	68.48	76.22
	BS	37.60	3.33	6.67	68.87	71.95
	RS	45.48 [158/500]	0.00 [6/30]	0.00 [4/30]	69.77 [250/257]	73.03 [163/164]
	SNIS	40.38	3.33	0.00	68.58	72.67
	MH-BS	35.40	0.00	6.67	68.48	73.17
Olmo-2-7B-SFT	AR	12.29	0.38	0.17	29.40	20.62
	MH	17.40	0.00	0.00	48.25	38.41
	BS	9.40	0.00	0.00	42.02	31.71
	RS	14.16 [180/500]	0.00 [4/30]	0.00 [5/30]	41.84 [257/257]	31.90 [162/164]
	SNIS	10.08	0.00	0.00	41.88	31.57
	MH-BS	8.80	0.00	0.00	40.86	29.88

4.2 Analysis

Accuracy varies across models and benchmarks but is broadly consistent across inference methods. Table 1 shows benchmark accuracy across all models and benchmarks we experimented with. For each model and benchmark, the table shows the accuracy under autoregressive sampling, beam search, and the rejection-sampling, importance-sampling, and Metropolis–Hastings procedures described above. The table shows that the effects of the power distribution depend strongly on the model and benchmark, but rejection sampling, importance sampling, and Metropolis–Hastings from the beam-search output often yield similar accuracy within a given setting. For example, all three methods substantially reduce Qwen2.5-7B's accuracy on math benchmarks while improving it on MBPP. Figure 5 shows that these results also hold when the set of benchmark problems is restricted only to those where rejection sampling produces a sample. This suggests that restricting the results in Table 1 to to only include the problems where a sample was produced with rejection sampling does not introduce a bias in our results. This agreement extends beyond accuracy: Figure 2

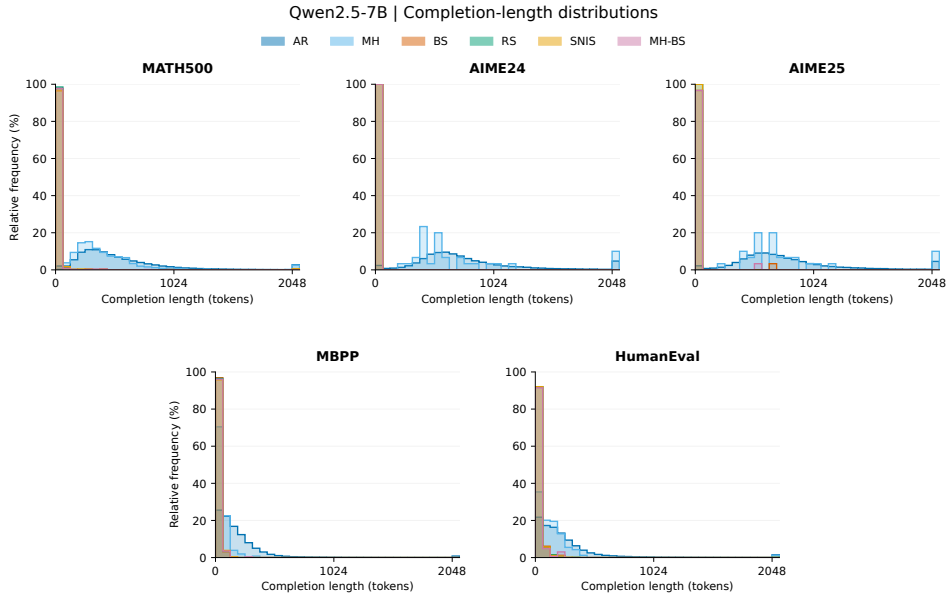


Figure 2: **Completion length under the power distribution.** Completion-length distributions for Qwen2.5-7B under autoregressive sampling (AR), Metropolis–Hastings from the empty completion (MH), beam search (BS), rejection sampling (RS), self-normalized importance sampling (SNIS), and Metropolis–Hastings from the beam-search output (MH-BS). MH, BS, and MH-BS each contribute a single output per problem.

shows similar completion-length distributions for Qwen2.5-7B across the three methods, with substantially more mass on short completions than under autoregressive sampling, and Figure 3 shows that the fraction of completions with length ≤ 5 tokens is similar across the three inference methods but differs significantly for autoregressive decoding. Agreement across distinct inference procedures and multiple distributional statistics supports interpreting these patterns as properties of the power distribution rather than as mere artifacts of approximate inference. The rows for Metropolis–Hastings from the empty completion use the same sampler but start each chain from the empty completion, as in prior work.

Accuracy under the power distribution largely matches that of beam search. Table 1 shows that accuracy under the power distribution is often close to that of beam search. Figure 1 further shows that the importance-sampling accuracy estimates change most sharply near $\alpha = 1$ and largely level off for $\alpha > 2$, suggesting that modest sharpening is sufficient for the power distribution to exhibit many of the same unstable behaviors that have been observed with beam search decoding. Table 2 shows that the methods for sampling from the power distribution assign substantially more mass to the completion found by best-first beam search. These results suggest that finite-power sampling already favors many of the same high-probability completions as beam search.

The power distribution up-weights degenerate completions. Figure 3 shows that, for Qwen2.5-7B, completions of at most five tokens account for 87–92% of the mass on MATH500 under rejection sampling, importance sampling, and Metropolis–Hastings from the beam-search output, compared with less than 1% under autoregressive sampling, and that almost none of them are correct. This concentration explains most of the decline in accuracy from roughly 45% to 5% in Table 1. The effect varies across models: on MATH500, short completions account for 6–37% of the mass for the other three models (Table 3), and for Qwen3-8B-Base many of them are correct (Table 4), consistent with the smaller change in its accuracy.

These results are a consequence of the similarity of the power distribution to beam search. Prior work has shown that the highest-likelihood completions are unstable, often exhibiting degeneracy and a short length bias. (Holtzman et al., 2020; Stahlberg & Byrne, 2019; Yoshida et al., 2024).

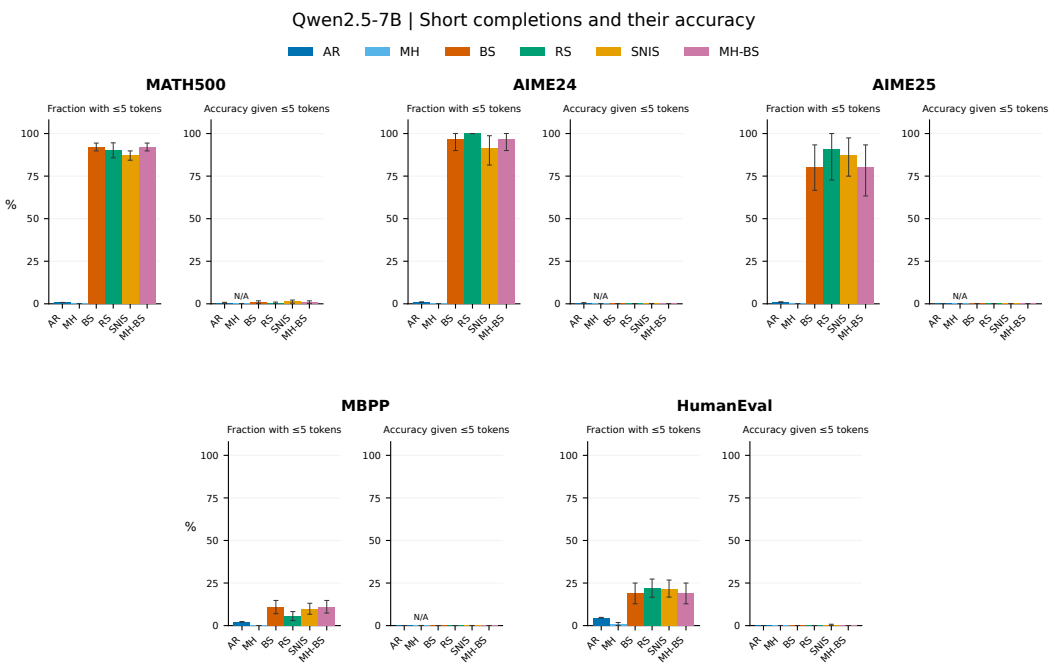


Figure 3: **Fraction (%) of short completions (less than or equal to five tokens) and accuracy given less than or equal to five tokens (%).** Short completions and their accuracy for Qwen2.5-7B under autoregressive sampling (AR), Metropolis–Hastings from the empty completion (MH), beam search (BS), rejection sampling (RS), self-normalized importance sampling (SNIS), and Metropolis–Hastings from the beam-search output (MH-BS). For each benchmark, the left panel shows the fraction of completions with at most five tokens; the right shows accuracy conditioned on that same length threshold.

At $\alpha = \infty$ the power distribution places all of its mass on the highest-likelihood completions. Our experiments suggest that at $\alpha = 4$ it already shifts much of its mass toward them. Figure 4 (Appendix E) illustrates why these completions are often degenerate. For the prompt "How many positive whole-number divisors does 196 have?", most of the probability mass of Qwen2.5-7B lies on correct solutions, but that mass is divided among many distinct phrasings of the solution, none of which is individually probable (Holtzman et al., 2021). The most probable completion, "ôt", begins with an improbable token but ends immediately with `EOS`, and so avoids accumulating a lower likelihood with each sampled non-`EOS` token.

Sequence lengths get shorter. In addition to increasing the frequency of degenerate completions with 5 of fewer tokens, Figure 2 shows that Qwen2.5-7B completions get shorter on average across all five benchmarks we experimented with. Autoregressive sampling produces a broad range of completion lengths, whereas rejection sampling, importance sampling, and Metropolis–Hastings from the beam-search output concentrate most of their mass in the shortest length bin. This shift occurs across all five benchmarks, indicating that global sharpening consistently favors shorter completions in these experiments. This observation contrasts RL, which has the tendency of lengthening completions (Shao et al., 2024).

Initializing Metropolis–Hastings at the beam-search output makes it resemble rejection and importance sampling. We observe that in Table 1 and Figures 2, 3, 5, that the distributional statistics of the Metropolis–Hastings sampler deviates significantly from those of rejection sampling and importance sampling when it is initialized from the empty string, but they match closely when the sampler is initialized at the completion produced by beam search. Especially notable is that, while rejection sampling and importance sampling often hurt benchmark accuracy, Metropolis–Hastings with empty initialization nearly always improves accuracy across models and benchmarks.

Because the power distribution with $\alpha = 4$ resembles beam search, this may be treated as initializing Metropolis–Hastings on a completion produced by a Monte Carlo chain that has already converged. We conclude from this that the initialization of Metropolis–Hastings plays a key role in its improving accuracy across benchmarks and models.

5 Related Work

Power sampling. Karan & Du (2026) introduce the power distribution as a training-free target for reasoning and approximate it with Metropolis–Hastings. Subsequent work proposes more efficient approximations based on sequential Monte Carlo and related methods (Azizi et al., 2026; Ji et al., 2026; Meng et al., 2026; Zhou et al., 2026) or amortizes the power distribution through fine-tuning (Chen et al., 2026). These methods are similar in their use of an inference scheme to draw approximate samples from an exponentiated version of the base distribution.

Distribution sharpening and RLVR. Several studies argue that RLVR primarily reweights outputs that the base model can already produce. Yue et al. (2025) find that RLVR improves pass@1 while base models attain higher pass@k at large k, Shao et al. (2025) report that RLVR with random or incorrect rewards can still improve accuracy for some model families, and He et al. (2025) show that GRPO preferentially reinforces already-probable solutions. Other prior work has found evidence against the distribution sharpening hypothesis as well. In a controlled chess setting, Shen et al. (2026) estimate an effective power coefficient α by regressing the centered log-probabilities of a GRPO-trained move policy on those of the supervised policy from which it was initialized, since exact power sharpening would make this relationship linear with slope α. The fitted slope increases during RL, but the fit explains only part of the variance and the per-state slopes vary substantially, indicating that RL is not well described by a single power transformation of the initial policy.

Most similar to our work is Mittal et al. (2026), who compare sharpening-based and task-reward-based objectives within a single RL fine-tuning framework. They fine-tune policies toward two sharpened objectives: one maximized by the argmax completion and one maximized by the power distribution. The power distribution enters their study only through policies fine-tuned toward the second objective. Because this policy is trained by stochastic, non-convex optimization and need not reach the optimum of its objective, their findings reflects the training algorithm as well as the target. These policies often reach accuracies that differ substantially from those of the first objective, even at values of α at which we find the power distribution to closely resemble the argmax completion (Section 4.2); for Qwen2.5-3B on MATH500, for example, the final checkpoint reaches 61.3% at $\alpha = 2$ against 19.6% for the first objective. Because such policies are trained by stochastic, non-convex optimization and need not reach the optimum of their objective, their behavior reflects the optimization as well as the target. We study the power distribution without training: rejection sampling yields exact samples wherever an argmax completion can be certified, which isolates properties of the power distribution from artifacts of both test-time approximate inference and RL fine-tuning.

6 Conclusion

In this paper, we investigate the hypothesis that the power distribution can explain reasoning gains associated with RLVR post-training. We probe the power distribution with a variety of techniques and find that even with moderate global tempering (low $\alpha > 1$), results closely resemble likelihood-maximization decoding (equivalent to the power distribution with $\alpha = \infty$). In particular, we observe that the accuracy under the power distribution closely matches that of beam search decoding—improving for some models and benchmarks and markedly degrading on others—the distribution of sequence lengths for the power distribution and beam search decoding are nearly identical, and the frequency of degenerate completions increases under both. Based on these findings, we conclude that the power distribution does not recover RLVR-like behaviors. Thus, prior methods that showed improved benchmark performance when sampling from the power distribution benefitted from artifacts of their approximate inference algorithm. What latent properties these sampling algorithms exploited remains an worthwhile open question, as these properties may be leveraged for developing verifier-free decoding strategies that improve base model benchmark accuracy.

AI USE STATEMENT

In this work, we used generative AI tools to assist with implementing portions of our experiments and to review the presentation of our methodology and experiments. We additionally used generative AI tools to edit the manuscript for clarity and concision and to revise LaTeX formatting and producing figures. We did not use generative AI to generate experimental data or conduct experiments. All AI-assisted code and manuscript revisions were reviewed by the authors; the code was tested, experiments were run by the authors, and reported values were checked against the underlying results. We take responsibility for the final content of this work, including all AI-assisted text, claims, code, and artifacts.

REFERENCES

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Ruishuo Chen, Yu Chen, Zhuoran Li, and Longbo Huang. Powerflow: Unlocking the dual nature of llms via principled distribution matching. *CoRR*, abs/2603.18363, 2026. doi: 10.48550/ARXIV.2603.18363. URL https://doi.org/10.48550/arXiv.2603.18363.

DeepSeek. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Bryan Eikema and Wilker Aziz. Is MAP decoding all you need? the inadequacy of the mode in neural machine translation. In Donia Scott, Núria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pp. 4506–4520. International Committee on Computational Linguistics, 2020. doi: 10.18653/V1/2020.COLING-MAIN.398. URL https://doi.org/10.18653/v1/2020.coling-main.398.

Andre Wang He, Daniel Fried, and Sean Welleck. Rewarding the unlikely: Lifting GRPO beyond distribution sharpening. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 25548–25560. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.1298. URL https://doi.org/10.18653/v1/2025.emnlp-main.1298.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. Surface form competition: Why the highest probability answer isn't always right. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 7038–7051. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.564. URL https://doi.org/10.18653/v1/2021.emnlp-main.564.

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. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=WJaUkwci9o.

Xiaotong Ji, Rasul Tutunov, Matthieu Zimmer, and Haitham Bou- Ammar. Scalable power sampling: Unlocking efficient, training-free reasoning for llms via distribution sharpening. *CoRR*, abs/2601.21590, 2026. doi: 10.48550/ARXIV.2601.21590. URL https://doi.org/10.48550/arXiv.2601.21590.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL https://arxiv.org/abs/2001.08361.

Aayush Karan and Yilun Du. Reasoning with sampling: Your base model is smarter than you think. In Carl Vondrick, Bharath Hariharan, Colin Raffel, Lerrel Pinto, Diyi Yang, and Aleksandra Faust (eds.), *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23-27, 2026*. proceedings.iclr.cc, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/01498f17142547dadb60999c39bd0a09-Abstract-Conference.html.

Mengqi Li, Lei Zhao, Anthony Man-Cho So, Ruoyu Sun, and Xiao Li. A model can help itself: Reward-free self-training for LLM reasoning. *CoRR*, abs/2510.18814, 2025. doi: 10.48550/ARXIV.2510.18814. URL https://doi.org/10.48550/arXiv.2510.18814.

Clara Meister, Ryan Cotterell, and Tim Vieira. Best-first beam search. *Trans. Assoc. Comput. Linguistics*, 8:795–809, 2020. doi: 10.1162/TACL_A_00346. URL https://doi.org/10.1162/tacl_a_00346.

Zibin Meng, Peng Xie, and Kani Chen. Depth-entropy guided sampling for training-free llm reasoning, 2026. URL https://arxiv.org/abs/2607.09693.

Sarthak Mittal, Léo Gagnon, and Guillaume Lajoie. Beyond distribution sharpening: The importance of task rewards. *CoRR*, abs/2604.16259, 2026. doi: 10.48550/ARXIV.2604.16259. URL https://doi.org/10.48550/arXiv.2604.16259.

OpenAI. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

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. *CoRR*, abs/2506.10947, 2025. doi: 10.48550/ARXIV.2506.10947. URL https://doi.org/10.48550/arXiv.2506.10947.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Jingyan Shen, Ang Li, Salman Rahman, Yifan Sun, Micah Goldblum, Matus Telgarsky, and Pavel Izmailov. Understanding reasoning from pretraining to post-training. *CoRR*, abs/2607.16097, 2026. doi: 10.48550/ARXIV.2607.16097. URL https://doi.org/10.48550/arXiv.2607.16097.

Felix Stahlberg and Bill Byrne. On NMT search errors and model errors: Cat got your tongue? In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3356–3362, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1331. URL https://aclanthology.org/D19-1331/.

John von Neumann. Various techniques used in connection with random digits. In *Monte Carlo Method*, volume 12 of *National Bureau of Standards Applied Mathematics Series*, pp. 36–38. U.S. Government Printing Office, Washington, DC, 1951. URL https://mcnp-green.lanl.gov/pdf_files/InBook_Computing_1961_Neumann_JohnVonNeumannCollectedWorks_VariousTechniquesUsedinConnectionwithRandomDigits.pdf. Summary written by George E. Forsythe.

Davis Yoshida, Kartik Goyal, and Kevin Gimpel. Map's not dead yet: Uncovering true language model modes by conditioning away degeneracy. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 16164–16215. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.855. URL https://doi.org/10.18653/v1/2024.acl-long.855.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan- Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico , November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/537d5aa768c2d534016a4d06f87bc8fb-Abstract-Conference.html.

Felix Zhou, Anay Mehrotra, and Quanquan C. Liu. Reasoning with sampling: Cutting at decision points. *CoRR*, abs/2605.30327, 2026. doi: 10.48550/ARXIV.2605.30327. URL https://doi.org/10.48550/arXiv.2605.30327.

A Naive Rejection Sampling and Its Acceptance Rate

Fix a prompt $\boldsymbol{x}$ and suppress conditioning on it. Let p be the distribution over completions ending at EOS or length T, and write the power distribution as $p_\alpha(\boldsymbol{y}) = p(\boldsymbol{y})^\alpha / Z_\alpha$, where $Z_\alpha = \sum_{\boldsymbol{y}} p(\boldsymbol{y})^\alpha$ and $\alpha > 1$. A rejection sampler is immediate because every sequence probability lies in $[0, 1]$, so

$$p(\boldsymbol{y})^\alpha \le p(\boldsymbol{y}). \tag{6}$$

Thus, using the base model p as the proposal, $M = 1$ is a valid envelope constant for the *unnormalized* target p^α. Draw a completion $\boldsymbol{y} \sim p$ by ordinary autoregressive sampling and accept it with probability $p(\boldsymbol{y})^{\alpha-1}$. This construction requires only sampling from the base model and evaluating the probability of the sampled sequence; it requires neither the normalizing constant nor an argmax search. These readily available ingredients make the naive sampler easy to derive and implement.

The accepted samples have the desired distribution: the probability of proposing and accepting $\boldsymbol{y}$ is $p(\boldsymbol{y})p(\boldsymbol{y})^{\alpha-1} = p(\boldsymbol{y})^\alpha$, which normalizes to p_α. The difficulty is its acceptance rate,

$$A_{\text{naive}} = \mathbb{E}_{\boldsymbol{y}\sim p}\left[p(\boldsymbol{y})^{\alpha-1}\right] = Z_\alpha, \qquad \mathbb{E}[\text{proposals per accepted sample}] = \frac{1}{Z_\alpha}. \tag{7}$$

Sequence probabilities multiply token probabilities across the entire completion, so even plausible completions can have very small individual probability. Raising these probabilities to the power $\alpha - 1$ makes acceptance extremely unlikely. More generally, if $p_{\max} = \max_{\boldsymbol{y}} p(\boldsymbol{y})$, then

$$A_{\text{naive}} = \sum_{\boldsymbol{y}} p(\boldsymbol{y})p(\boldsymbol{y})^{\alpha-1} \le p_{\max}^{\alpha-1}. \tag{8}$$

For illustration, at $\alpha = 4$, even a maximum sequence probability of 10^{-2} implies an acceptance rate of at most 10^{-6}, or at least one million proposals per accepted sample in expectation. This is an illustrative bound, not an empirical estimate; the rate can be much smaller when probability mass is spread across many completions.

The role of the certified argmax in Algorithm 1 is to tighten this loose envelope. It permits $M = p_{\max}^{\alpha-1}$ instead of $M = 1$, giving acceptance rate $Z_\alpha / p_{\max}^{\alpha-1}$ and improving acceptance by a factor of $p_{\max}^{-(\alpha-1)}$. This does not guarantee a high absolute acceptance rate for every prompt, but removes the avoidable penalty from the naive envelope. The obstacle with $M = 1$ is therefore not constructing a correct sampler: it is obtaining any accepted samples within a practical proposal budget.

B Correctness of the Rejection Sampler

Correctness follows by applying classical rejection sampling (von Neumann, 1951). Fix a prompt $\boldsymbol{x}$ and suppress conditioning on it. Let p be the distribution over completions ending at EOS or length T. A valid argmax certificate for $\hat{\boldsymbol{y}}$ gives $p(\boldsymbol{y}) \le p(\hat{\boldsymbol{y}})$ for every completion. For $\alpha > 1$, this implies

$$p(\boldsymbol{y})^\alpha \le p(\hat{\boldsymbol{y}})^{\alpha-1} p(\boldsymbol{y}) = M p(\boldsymbol{y}), \qquad M = p(\hat{\boldsymbol{y}})^{\alpha-1}. \tag{9}$$

Thus Mp is an upper bound on the unnormalized target p^α. Rejection sampling with proposal p therefore accepts each proposal $\boldsymbol{y}$ with probability

$$\frac{p(\boldsymbol{y})^\alpha}{M p(\boldsymbol{y})} = \left(\frac{p(\boldsymbol{y})}{p(\hat{\boldsymbol{y}})}\right)^{\alpha-1}, \tag{10}$$

which is exactly the rule in Algorithm 1. The normalizing constant Z_α is not needed.

Under exact probability evaluation, independent proposals and uniform draws give independent accepted samples from p_α. The fixed budget of KB trials affects only the number returned: conditional on returning $n \ge 1$ samples, they are independent draws from p_α. If no valid certificate is obtained, the algorithm returns an empty list.

C Frequency of Matching Beam Search

Table 2 measures how often each method returns the saved best-first beam-search completion for the same problem, whether or not that search output has an argmax certificate. We compare exact decoded completion text, including whitespace, rather than extracted answers or correctness. We use text because the original Qwen2.5 Metropolis–Hastings records omit terminal EOS tokens retained in the best-first records. The reference is the particular output returned by beam search; an uncertified output need not be the true argmax.

Let $g_j(\boldsymbol{y}) = \mathbf{1}\{\mathrm{text}(\boldsymbol{y}) = \mathrm{text}(\hat{\boldsymbol{y}}_j^{\mathrm{BF}})\}$ for the saved beam-search completion $\hat{\boldsymbol{y}}_j^{\mathrm{BF}}$ on problem j. Autoregressive sampling averages this indicator over $1{,}024$ temperature-1 completions; rejection sampling averages over accepted proposals only; Metropolis–Hastings from the beam-search output uses its one final recorded completion. Importance sampling estimates the match probability as $\sum_i \bar{w}_{ji} g_j(\boldsymbol{y}_{ji})$, using the normalized $\alpha = 4$ weights from Equation 4. Importance sampling therefore reports estimated probability mass, not a frequency of separately generated samples. All reported percentages average the per-problem estimates equally.

Autoregressive sampling, importance sampling, and Metropolis–Hastings from the beam-search output use every problem in each benchmark, including problems whose beam-search outputs are uncertified. Only rejection sampling is restricted to problems with a saved argmax certificate and at least one accepted proposal, as in Table 1. Brackets beneath each rejection-sampling row report its included/full benchmark counts; these conditional rates are not directly comparable to the full-benchmark rates of the other methods. Some rejection-sampling subsets contain only one problem. Importance sampling assigns zero estimated mass to a beam-search completion absent from its finite proposal pool.

For Qwen2.5-7B on all 500 MATH500 problems, autoregressive sampling matches beam search 0.03% of the time, importance sampling assigns it 24.76% estimated mass, and Metropolis–Hastings from the beam-search output matches it 98.00% of the time. Rejection sampling has a 73.89% match rate on its qualifying subset of 165 problems. The high match rate of Metropolis–Hastings from the beam-search output measures agreement with its best-first initialization; it does not by itself establish convergence to the power distribution.

Table 2: Exact decoded-text matches to the saved best-first beam-search output (%). The methods are autoregressive sampling (AR), rejection sampling (RS), self-normalized importance sampling (SNIS), and Metropolis–Hastings from the beam-search output (MH-BS). AR uses temperature 1; RS, SNIS, and MH-BS use $\alpha = 4$. AR, SNIS, and MH-BS use the full benchmark, including uncertified search outputs. Only RS is restricted to problems with a saved argmax certificate and at least one accepted proposal; brackets below RS give included/full benchmark problem counts. SNIS reports normalized importance-weight mass.

Model	Method	MATH500	AIME24	AIME25	MBPP	HumanEval
Qwen2.5-7B	AR	0.03	0.05	0.05	1.14	0.53
	RS	73.89	90.00	81.82	68.07	66.30
		[165/500]	[10/30]	[11/30]	[231/257]	[163/164]
	SNIS	24.76	34.13	42.53	61.55	65.93
	MH-BS	98.00	100.00	96.67	80.54	89.63
Qwen3-8B-Base	AR	0.03	0.00	0.01	4.02	2.24
	RS	65.79	100.00	100.00	74.98	64.00
		[71/500]	[1/30]	[1/30]	[252/257]	[161/164]
	SNIS	10.15	3.33	10.00	74.89	63.43
	MH-BS	35.80	26.67	30.00	80.16	78.05
Olmo-3-7B-SFT	AR	0.06	0.02	0.03	14.99	14.47
	RS	60.97	75.00	100.00	74.43	72.10
		[158/500]	[6/30]	[4/30]	[250/257]	[163/164]
	SNIS	19.45	13.27	12.58	72.74	71.92
	MH-BS	66.60	73.33	86.67	81.32	79.88
Olmo-2-7B-SFT	AR	0.06	0.01	0.02	6.53	4.92
	RS	67.75	75.00	100.00	75.69	72.26
		[180/500]	[4/30]	[5/30]	[257/257]	[162/164]
	SNIS	24.56	9.98	16.66	76.51	71.13
	MH-BS	75.20	76.67	70.00	85.21	84.15

D SHORT COMPLETIONS

Throughout, the length $\ell(\boldsymbol{y})$ of a completion counts completion tokens, excluding the prompt and the terminal EOS, so an immediate EOS has length zero. Completions that reach the generation limit are retained.

D.1 COMPLETIONS OF AT MOST FIVE TOKENS

Table 3 reports the fraction of completions with $\ell(\boldsymbol{y}) \leq 5$ for all four models. The threshold includes empty completions and excludes the prompt and terminal EOS. For autoregressive sampling and rejection sampling, we average the indicator $\mathbf{1}\{\ell(\boldsymbol{y}) \leq 5\}$ over the available completions within each problem; for Metropolis–Hastings from the beam-search output, we evaluate its final completion. Importance sampling estimates the probability of a short completion as $\sum_i \bar{w}_{ji} \mathbf{1}\{\ell(\boldsymbol{y}_{ji}) \leq 5\}$. We then average equally across included problems and report percentages. Only rejection sampling is restricted to problems with a saved argmax certificate and at least one accepted proposal, with counts shown beneath its rows.

Table 3: Completions with at most five tokens (%), excluding the prompt and terminal EOS. The methods are autoregressive sampling (AR), rejection sampling (RS), self-normalized importance sampling (SNIS), and Metropolis–Hastings from the beam-search output (MH-BS). AR uses temperature 1; RS, SNIS, and MH-BS use $\alpha = 4$. AR, SNIS, and MH-BS use the full benchmark; brackets below RS give qualifying/full benchmark problem counts. SNIS reports normalized importance-weight mass on completions of length ≤ 5.

Model	Method	MATH500	AIME24	AIME25	MBPP	HumanEval
Qwen2.5-7B	AR	0.77	0.97	1.00	2.24	4.64
	RS	90.40	100.00	90.91	5.46	21.92
		[165/500]	[10/30]	[11/30]	[231/257]	[163/164]
	SNIS	87.17	91.18	87.47	9.76	21.66
	MH-BS	92.20	96.67	80.00	10.89	18.90
Qwen3-8B-Base	AR	0.03	0.06	0.07	0.39	0.92
	RS	19.48	0.00	0.00	0.26	4.55
		[71/500]	[1/30]	[1/30]	[252/257]	[161/164]
	SNIS	8.51	33.45	19.88	0.57	4.36
	MH-BS	6.00	6.67	6.67	0.39	5.49
Olmo-3-7B-SFT	AR	0.07	0.24	0.14	0.01	0.08
	RS	20.09	41.67	25.00	0.00	0.00
		[158/500]	[6/30]	[4/30]	[250/257]	[163/164]
	SNIS	15.63	38.35	43.90	0.00	0.00
	MH-BS	37.00	76.67	73.33	0.39	0.00
Olmo-2-7B-SFT	AR	0.15	0.07	0.07	1.79	0.61
	RS	19.94	50.00	100.00	22.18	3.22
		[180/500]	[4/30]	[5/30]	[257/257]	[162/164]
	SNIS	18.71	29.79	41.89	22.02	4.06
	MH-BS	25.60	60.00	60.00	22.57	3.05

D.2 Accuracy of Completions of at Most Five Tokens

Table 4 reports accuracy conditioned on completion length $\ell(\boldsymbol{y}) \leq 5$. This inclusive cutoff includes lengths zero through five and matches the ≤ 5 cutoff in Table 3. Grading uses the same saved correctness labels as Table 1.

For each problem, autoregressive sampling and rejection sampling compute accuracy among their short completions, using only accepted proposals for rejection sampling. Importance sampling renormalizes its $\alpha = 4$ importance weights over short proposals before computing weighted correctness. Metropolis–Hastings from the beam-search output contributes its final completion only if it is short. We then average equally over problems with qualifying outputs, rather than pooling completions across problems. Rejection sampling additionally requires a saved argmax certificate. Because the length restriction changes every method's coverage, brackets report included/full benchmark problem counts for all rows. A dash indicates that no completion qualifies; it does not indicate zero accuracy.

Table 4: Accuracy (%) conditioned within each problem on completion length ≤ 5 tokens, excluding the prompt and terminal EOS, then averaged equally over included problems. The methods are autoregressive sampling (AR), rejection sampling (RS), self-normalized importance sampling (SNIS), and Metropolis–Hastings from the beam-search output (MH-BS). AR uses temperature 1; RS, SNIS, and MH-BS use $\alpha = 4$. Brackets give included/full benchmark problem counts for every method. A dash denotes no qualifying completion. SNIS renormalizes importance weights over short proposals. RS additionally requires a saved argmax certificate and an accepted short proposal.

Model	Method	MATH500	AIME24	AIME25	MBPP	HumanEval
Qwen2.5-7B	AR	0.49 [496/500]	0.22 [30/30]	0.00 [30/30]	0.00 [257/257]	0.03 [164/164]
	RS	0.33 [150/500]	0.00 [10/30]	0.00 [10/30]	0.00 [20/257]	0.00 [62/164]
	SNIS	1.20 [496/500]	0.07 [30/30]	0.00 [30/30]	0.00 [257/257]	0.27 [164/164]
	MH-BS	0.87 [461/500]	0.00 [29/30]	0.00 [24/30]	0.00 [28/257]	0.00 [31/164]
Qwen3-8B-Base	AR	12.68 [116/500]	0.00 [13/30]	0.00 [13/30]	0.00 [243/257]	0.65 [163/164]
	RS	71.43 [14/500]	— [0/30]	— [0/30]	0.00 [2/257]	8.33 [12/164]
	SNIS	12.94 [116/500]	0.00 [13/30]	0.00 [13/30]	0.00 [243/257]	0.69 [163/164]
	MH-BS	70.00 [30/500]	0.00 [2/30]	0.00 [2/30]	0.00 [1/257]	11.11 [9/164]
Olmo-3-7B-SFT	AR	9.00 [170/500]	0.00 [16/30]	0.00 [19/30]	0.00 [17/257]	2.02 [24/164]
	RS	38.16 [38/500]	0.00 [3/30]	0.00 [1/30]	— [0/257]	— [0/164]
	SNIS	12.27 [170/500]	0.00 [16/30]	0.00 [19/30]	0.00 [17/257]	0.01 [24/164]
	MH-BS	4.86 [185/500]	0.00 [23/30]	0.00 [22/30]	0.00 [1/257]	— [0/164]
Olmo-2-7B-SFT	AR	3.02 [310/500]	0.00 [13/30]	0.00 [16/30]	0.00 [251/257]	0.98 [142/164]
	RS	3.77 [42/500]	0.00 [2/30]	0.00 [5/30]	0.00 [88/257]	0.00 [11/164]
	SNIS	3.68 [310/500]	0.00 [13/30]	0.00 [16/30]	0.00 [251/257]	1.16 [142/164]
	MH-BS	5.47 [128/500]	0.00 [18/30]	0.00 [18/30]	0.00 [58/257]	0.00 [5/164]

E PROBABILITY MASS OF CORRECT AND DEGENERATE COMPLETIONS

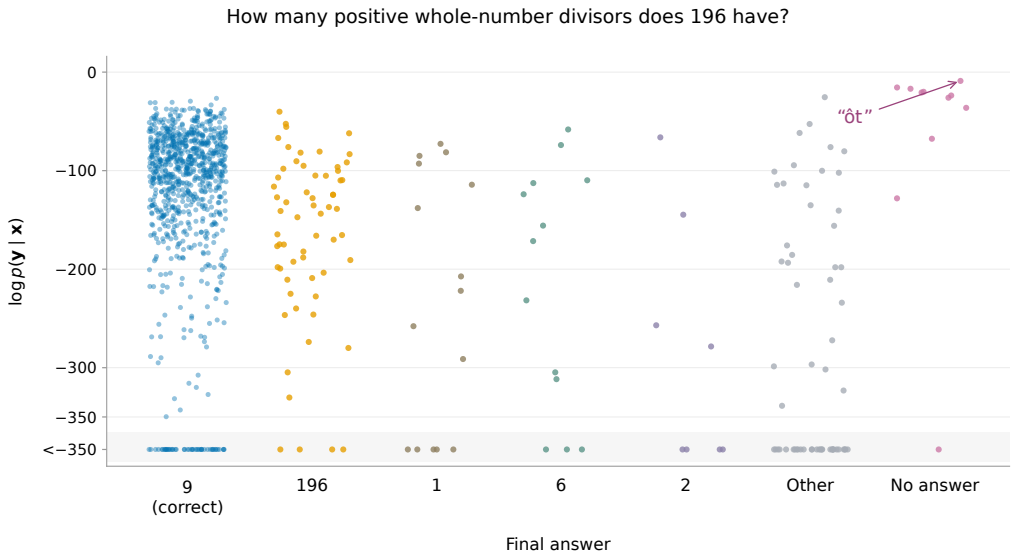


Figure 4: **Individual sequence probabilities for** 1,024 **temperature-1 Qwen2.5-7B completions.** Each point is a completion, positioned by its final answer and full-sequence log-likelihood. Many completions answer 9 correctly, while rare degenerate outputs have higher individual probability; the annotation identifies the most probable observed completion, "ôt". Horizontal jitter separates points. "Other" contains less frequent answers, and "No answer" denotes outputs without an extractable answer. The shaded bottom strip retains individual points with $\log p(\boldsymbol{y} \mid \boldsymbol{x}) < -350$.

F METROPOLIS–HASTINGS IMPLEMENTATION

At each stage of Algorithm 2, Metropolis–Hastings updates operate on completions that end at EOS or the current length cap. We derive the acceptance rule for one fixed cap, then explain what this guarantees for the staged procedure. Conditioning on the prompt is suppressed throughout.

Target and proposal at a fixed cap. Let $\mathcal{Y}_L$ contain all completions that emit EOS by token L, together with all length-L outputs that have not emitted EOS. Define

$$\pi_L(\boldsymbol{y}) = \frac{p(\boldsymbol{y})^\alpha}{\sum_{\boldsymbol{z}\in\mathcal{Y}_L} p(\boldsymbol{z})^\alpha}, \qquad \boldsymbol{y} \in \mathcal{Y}_L. \tag{11}$$

Here p includes the EOS probability for terminated completions and includes no artificial EOS factor for capped outputs. This is a power distribution on the capped output space, not generally the marginal of the final-horizon power distribution. We write $|\boldsymbol{y}|$ for the number of tokens before EOS, matching the length convention used elsewhere in the paper.

Given a current completion $\boldsymbol{y}$ with $n = |\boldsymbol{y}| \geq 1$, choose a cut index $m \in \{1, \ldots, n\}$ uniformly. Retain $\boldsymbol{y}_{<m}$ and draw a new suffix from p_{prop}, starting at position m, until EOS or the same cap L. The suffix may be longer or shorter than the current one. For a proposal $\boldsymbol{y}'$ with $n' = |\boldsymbol{y}'|$, write its suffix probability, including the EOS token when present, as

$$q_{m,L}(\boldsymbol{y}' \mid \boldsymbol{y}_{<m}) = \prod_{t\geq m} p_{\text{prop}}(y'_t \mid \boldsymbol{y}'_{<t}). \tag{12}$$

The proposal is compatible with the cut if $\boldsymbol{y}' \in \mathcal{Y}_L$, $\boldsymbol{y}'_{<m} = \boldsymbol{y}_{<m}$, and $n' \geq m$, that is, if the new suffix contains at least one token before EOS. An incompatible proposal is rejected, because the reverse move from $\boldsymbol{y}'$ could not select the cut m. We assume the proposal has positive probability for every token on the target support. Pair each forward proposal with the reverse proposal using the same cut index m; that index is valid in both completions. The joint probabilities of selecting the index and proposing the completion are

$$Q_m(\boldsymbol{y}, \boldsymbol{y}') = \frac{q_{m,L}(\boldsymbol{y}' \mid \boldsymbol{y}_{<m})}{n}, \qquad Q_m(\boldsymbol{y}', \boldsymbol{y}) = \frac{q_{m,L}(\boldsymbol{y} \mid \boldsymbol{y}_{<m})}{n'}. \tag{13}$$

These are probabilities for an indexed proposal, not the complete transition probability: multiple cut indices can yield the same pair of completions.

Acceptance rule and detailed balance. For the sampled index, use

$$a_m(\boldsymbol{y}, \boldsymbol{y}') = \min\left\{1, \frac{\pi_L(\boldsymbol{y}')}{\pi_L(\boldsymbol{y})} \frac{n}{n'} \frac{q_{m,L}(\boldsymbol{y} \mid \boldsymbol{y}_{<m})}{q_{m,L}(\boldsymbol{y}' \mid \boldsymbol{y}_{<m})}\right\}. \tag{14}$$

The numerator contains the reverse suffix probability. The factor n/n' accounts for the different probabilities of selecting m in the two states. For each compatible index,

$$\begin{aligned} \pi_L(\boldsymbol{y}) Q_m(\boldsymbol{y}, \boldsymbol{y}') a_m(\boldsymbol{y}, \boldsymbol{y}') &= \min\{\pi_L(\boldsymbol{y}) Q_m(\boldsymbol{y}, \boldsymbol{y}'), \pi_L(\boldsymbol{y}') Q_m(\boldsymbol{y}', \boldsymbol{y})\} \\ &= \pi_L(\boldsymbol{y}') Q_m(\boldsymbol{y}', \boldsymbol{y}) a_m(\boldsymbol{y}', \boldsymbol{y}). \end{aligned} \tag{15}$$

Summing over the compatible indices establishes detailed balance for the completion-valued kernel; rejected proposals contribute self-transitions. Thus π_L is invariant. With full proposal support, choosing $m = 1$ allows any completion with $|\boldsymbol{y}| \geq 1$ to be proposed, and proposing the current completion gives a positive self-transition. On this finite space the chain is therefore irreducible and aperiodic on $\mathcal{Y}_L^+ = \{\boldsymbol{y} \in \mathcal{Y}_L : |\boldsymbol{y}| \geq 1\}$ and converges to π_L restricted to $\mathcal{Y}_L^+$.

The empty completion. The completion that consists only of EOS has no admissible cut. The sampler never proposes it, and a chain initialized at it cannot move and returns it unchanged. This case arises only under best-first initialization, when the best-first output is an immediate EOS. It occurs for 7 of 500 MATH500 problems for Olmo-3-7B-SFT, and for 33 of 500 MATH500, 13 of 30 AIME24, 14 of 30 AIME25, and 1 of 164 HumanEval problems for Olmo-2-7B-SFT. It does not occur for either Qwen model.

Algorithm 2: Metropolis-Hastings Sampling with Degenerate Initialization

Input : prompt $\boldsymbol{x}$; base p; proposal p_{prop}; power α; maximum length T

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

Output : sequence $\boldsymbol{y}$ approximating a draw from p_α

Notation: With conditioning on $\boldsymbol{x}$ suppressed, the intermediate target is

$$\pi_L(\boldsymbol{y}) \;\propto\; p(\boldsymbol{y})^\alpha, \qquad \boldsymbol{y} \in \mathcal{Y}_L.$$

Outputs end at EOS or cap L; $|\boldsymbol{y}|$ is the number of tokens before EOS. A chain with $|\boldsymbol{y}| = 0$ does not move, and a proposal whose new suffix has no token before EOS is rejected.

$(\hat{\boldsymbol{y}}, c_0) \leftarrow \text{BestFirst}(p, \boldsymbol{x}, T)$ `// Stage 1 of Algorithm 1`

$\boldsymbol{y} \leftarrow \hat{\boldsymbol{y}}$; $j_0 \leftarrow \max\{0, \lceil |\boldsymbol{y}|/S \rceil - 1\}$

for $j \leftarrow j_0$ **to** $\lceil \frac{T}{S} \rceil - 1$ **do**

  $L_j \leftarrow \min\{(j+1)S, T\}$

  if $\boldsymbol{y}$ has not ended at EOS **then** extend $\boldsymbol{y}$ with p_{prop} until EOS or length L_j

  Use the current state $\boldsymbol{y}$ as the initialization for π_{L_j}

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

    Sample an index $m \in \{1, \ldots, |\boldsymbol{y}|\}$ uniformly

    Construct $\boldsymbol{y}'$ by keeping $\boldsymbol{y}_{<m}$ and resampling the suffix:

$$y'_t \sim p_{\text{prop}}\big(\cdot \mid \boldsymbol{y}'_{<t}\big), \qquad t = m, m+1, \ldots$$

    Stop the proposal at EOS or length L_j

    Compute acceptance probability (Equation 14)

$$A(\boldsymbol{y}', \boldsymbol{y}) \;\leftarrow\; \min\left\{1,\; \frac{\pi_{L_j}(\boldsymbol{y}')}{\pi_{L_j}(\boldsymbol{y})} \cdot \frac{|\boldsymbol{y}|}{|\boldsymbol{y}'|} \cdot \frac{p_{\text{prop}}(\boldsymbol{y}_{m:} \mid \boldsymbol{y}_{<m})}{p_{\text{prop}}(\boldsymbol{y}'_{m:} \mid \boldsymbol{y}_{<m})}\right\}.$$

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

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

  end

end

return $\boldsymbol{y}$

Staged sampling. A proposed suffix may grow to the current cap even if the current completion ends earlier at EOS, and Metropolis–Hastings updates continue after the current completion ends at EOS. Both choices preserve the forward/reverse pairing used above.

In Algorithm 2, stage j uses cap $L_j = \min\{(j+1)S, T\}$ and target π_{L_j}. When the cap increases, an unterminated state is extended to EOS or the new cap to obtain a valid initializer. Such extension does not itself produce a draw from the next target. The argument above establishes invariance of the kernel at each fixed cap, not equilibrium after a finite number of updates or after changing caps. In particular, convergence to the final target requires sufficient mixing at cap T; ten updates per block do not provide a convergence guarantee.

For degenerate initialization, we start from the sequence returned by best-first search and retain the current state as the proposal length limit increases. We begin at the first block large enough to contain the initial sequence. The best-first output can be used as an initializer whether or not it is certified.

G Accuracy on Problems with Exact Samples

Figure 5 restricts the comparison to the problems on which rejection sampling accepted at least one proposal for Qwen2.5-7B, so that every method is evaluated against exact samples on the same problems.

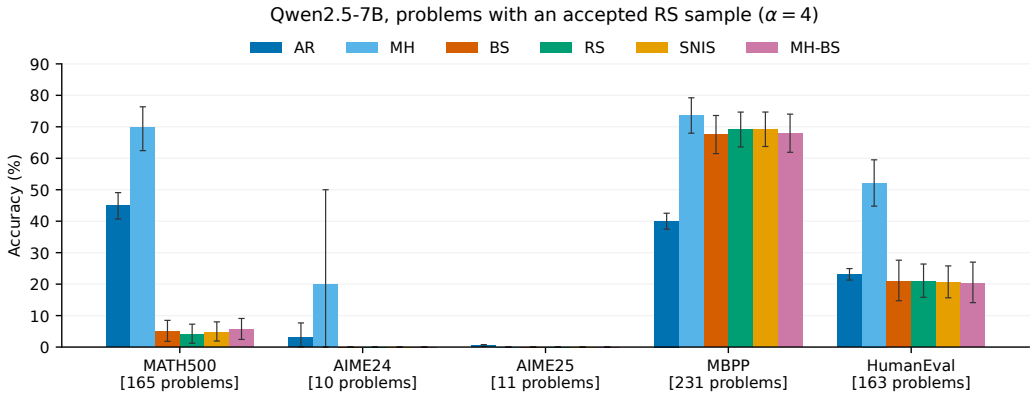


Figure 5: Accuracy (%) of Qwen2.5-7B at $\alpha = 4$ on the problems where rejection sampling accepted at least one proposal; brackets give the number of such problems. The methods are autoregressive sampling (AR), Metropolis–Hastings from the empty completion (MH), beam search (BS), rejection sampling (RS), self-normalized importance sampling (SNIS), and Metropolis–Hastings from the beam-search output (MH-BS). Error bars are 95% bootstrap intervals over problems (10,000 resamples, with the same resampled problems for every method); they do not include the randomness of the samplers.

H Experiment hyperparameters

We use the checkpoints `Qwen/Qwen2.5-7B`, `Qwen/Qwen3-8B-Base`, `allenai/Olmo-3-7B-Instruct-SFT`, and `allenai/OLMo-2-1124-7B-SFT`, referred to in the main text as Qwen2.5-7B, Qwen3-8B-Base, Olmo-3-7B-SFT, and Olmo-2-7B-SFT. All experiments use the following settings.

- bf16 precision and the standard chat template of each model.
- Autoregressive sampling draws 1,024 completions per problem at temperature 1 from the full vocabulary, without top-k or top-p truncation.
- Length limits. For Qwen2.5-7B, every method generates at most 2,048 tokens. For the other three models, autoregressive sampling and best-first search are limited to 4,096 prompt and completion tokens in total, so the completion budget is 4,096 minus the prompt length. Their Metropolis–Hastings runs generate at most 3,072 tokens, which fits within the 4,096-token context of every model for all prompts.
- Metropolis–Hastings uses $\alpha = 4$, a proposal equal to the base model at temperature $1/\alpha = 0.25$ without truncation, blocks of 192 tokens, and 10 Metropolis–Hastings steps per block. A chain started from the empty completion therefore runs 16 blocks and 160 Metropolis–Hastings steps at the 3,072-token limit, and 11 blocks and 110 Metropolis–Hastings steps at the 2,048-token limit for Qwen2.5-7B. A chain initialized at the best-first output starts at the first block that contains it and therefore runs fewer steps.

Our Metropolis–Hastings implementation is a corrected version of `github.com/maxzuo/mh-llm`; Appendix F describes it.

We note that our Qwen2.5-7B results are lower than on other benchmarks because we don't use Huggingface's default top-50 truncation.

The Qwen3 model is questionable, though, because it is trained on lots of synthetic data from post-trained models.

I ACCURACY VERSUS NUMBER OF PARTICLES

Figure 6 shows how the importance-sampling estimate changes as additional saved autoregressive completions are included, using the same fixed order for every particle count.

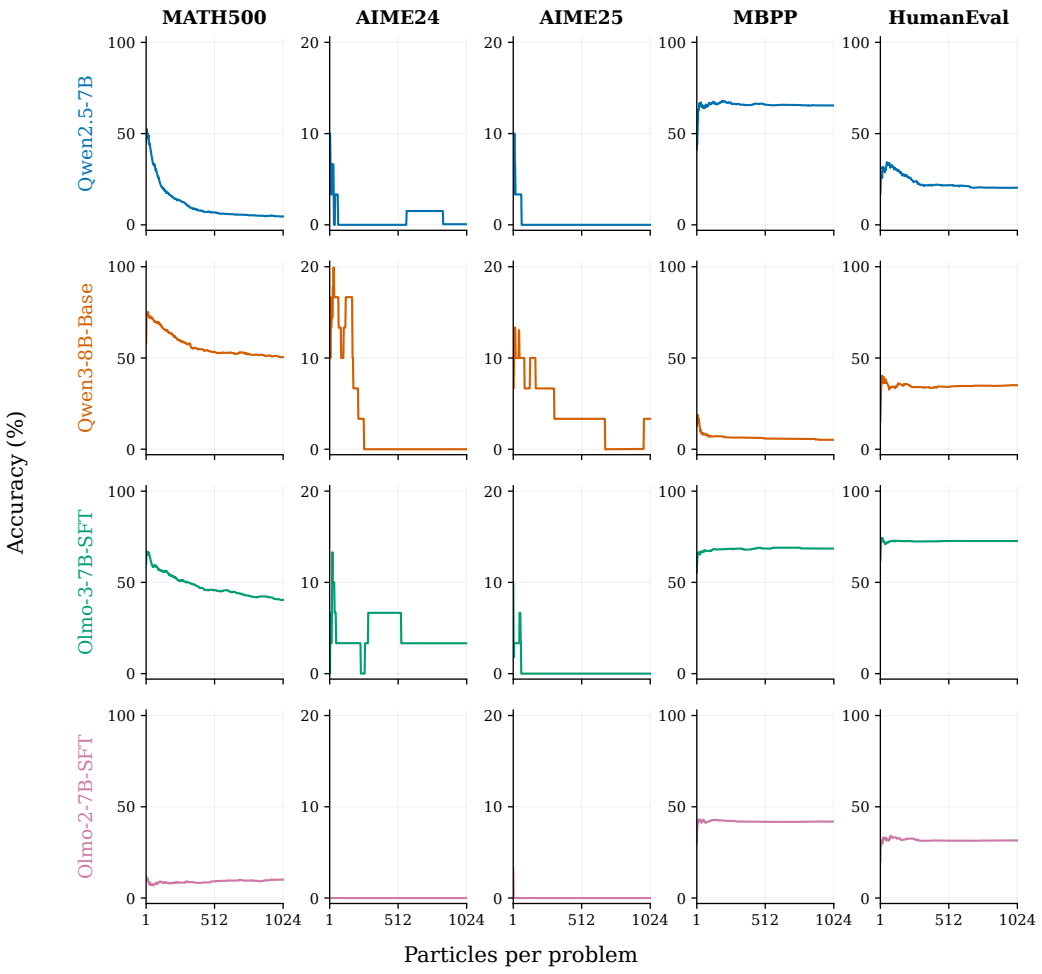


Figure 6: Importance-sampling estimates of accuracy at $\alpha = 4$ as the number of particles increases, for four models (rows) and five benchmarks (columns). For each problem, point N uses saved temperature-1 completions with indices $0, \ldots, N-1$, normalizing weights within the problem before averaging accuracy across the benchmark. Every N from 1 to 1,024 is plotted. AIME panels use a 0–20% vertical scale; other panels use 0–100%. Grading follows Table 1.

J Effective Sample Size of the Importance Weights

For each problem, the effective sample size of the self-normalized weights in Equation 4 is

$$\mathrm{ESS}_\alpha = \Big(\sum_{i=1}^{N} \bar{w}_i^2\Big)^{-1}, \qquad \bar{w}_i \propto p(\boldsymbol{y}_i \mid \boldsymbol{x})^{\alpha-1}, \tag{16}$$

which ranges from 1, when a single completion carries all the weight, to $N = 1{,}024$ at $\alpha = 1$. Figure 7 shows its distribution across problems. The effective sample size falls quickly as α grows. At $\alpha = 4$, its median across problems is between 1.00 and 1.24 on every mathematics benchmark, so the estimate for a typical mathematics problem is determined by the single most probable completion among the 1,024 proposals. On the code benchmarks, the median at $\alpha = 4$ is between 8.8 and 167.

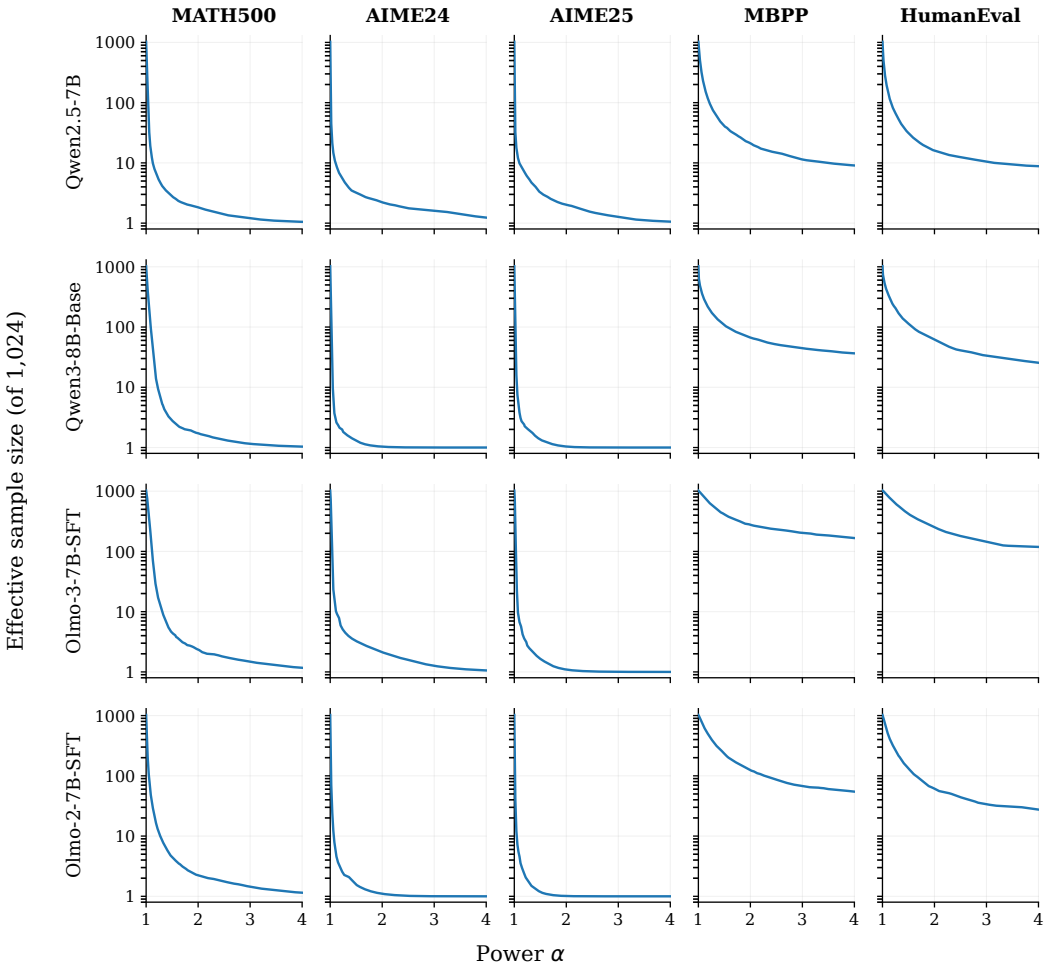


Figure 7: Effective sample size of the self-normalized importance weights as a function of α, for four models (rows) and five benchmarks (columns), using 1,024 temperature-1 completions per problem. Lines show the median across problems. The vertical axis is logarithmic.

Table 5: MATH500 problem 54 under all four samplers (Qwen2.5-7B, $\alpha = 4$). The methods are autoregressive sampling (AR), rejection sampling (RS), self-normalized importance sampling (SNIS), and Metropolis–Hastings from the beam-search output (MH-BS). The AR completion is a typical correct draw (the median log probability among the 896 correct completions in the pool). The SNIS row reports the normalized weight of the highest-weight completion; the RS row reports the single accepted proposal out of 1,024; the MH-BS row is a separate generation initialized at a certified best-first sequence.

Sampler	Completion	$\log p$	Tokens	Verdict
AR ($\alpha = 1$)	see below	-94.33	181	correct
SNIS	ô t ($\bar{w} = 1.0000$)	-8.03	2	no answer
RS	ô t	-8.03	2	no answer
MH-BS	ô t	—	1	no answer

K Example Samples

This appendix shows verbatim completions drawn by each of our samplers. All examples are Qwen2.5-7B on MATH500 with $\alpha = 4$, the setting of Figure **??**. Raw completion text is available locally only for this model, so we do not show examples from the other three models.

Importance sampling and rejection sampling both reweight the *same* pool of 1,024 saved temperature-1 completions per problem, so for a given problem their outputs are drawn from a common candidate set and are not independent of the autoregressive baseline. Metropolis–Hastings is generated separately and initialized at the best-first sequence. We report the full-sequence natural log probability $\log p(\boldsymbol{y} \mid \boldsymbol{x})$ under the base model, the completion length in tokens, and the Math-Verify (v0.9.0) verdict.

K.1 The Same Problem Under All Four Samplers

Table 5 shows MATH500 problem 54, "Write $\frac{3}{20}$ as a decimal." (reference answer 0.15). At $\alpha = 1$ the base model answers this problem correctly in 87.5% of 1,024 completions, with a mean length of 265 tokens. Only 4 of those 1,024 completions are shorter than five tokens, yet one of them carries essentially all of the mass at $\alpha = 4$: the self-normalized importance weights concentrate on a single completion (effective sample size 1.00), the rejection sampler accepts that same completion and nothing else, and Metropolis–Hastings independently converges to the same string. All three sharpened samplers return the two-token completion ô t.

We use problem 54 rather than the problem in Figure 4 because its saved argmax bound is not violated by any of its 1,024 proposals, so the acceptance test is applied with a valid envelope. The behavior is identical on the figure's problem, "How many positive whole-number divisors does 196 have?", where the base model is correct 83.6% of the time and all three sharpened samplers again return ô t; there, however, the accepted proposal's saved log probability exceeds its saved argmax bound by a small amount, which is the score-consistency limitation discussed in Section 4.

The base-model completion in the first row of Table 5 reads as follows.

> To convert the fraction $\frac{3}{20}$ to a decimal, we divide the numerator by the denominator:
>
> 1. Set up the division: $3 \div 20$
>
> 2. since 3 is smaller than 20, the first digit after the decimal point will be 0: 0.
>
> 3. Next, we will use a decimal point and add zeros to 3, making it 30. Divide 30 by 20, which gives us 1 with a remainder of 10: 0.1
>
> 4. Bring down another 0, making it 100. Divide 100 by 20, which gives us 5 with no remainder left after the 0th decimal place.
>
> So, the final result is 0.15.
>
> $\boxed{0.15}$

This is the qualitative content of Figure 4: the correct answer is spread over hundreds of distinct trajectories, each individually improbable (log probabilities from -5932.5 to -13.6 on this problem),

Table 6: Families of completions returned by the sharpened samplers (Qwen2.5-7B, MATH500, $\alpha = 4$), over the 216 accepted rejection-sampling proposals and the 500 Metropolis–Hastings completions. "Correct" counts the Math-Verify verdicts within each family. Strings are shown verbatim; `RoundedRectangle` has a leading space. Every correct completion falls in the last two rows.

		Count		Correct	
Family	Representative strings	RS	MH-BS	RS	MH-BS
Orphaned subword	`ôt`, `t`, `ivet`, `artz`, `rst`	135	346	0	0
Code identifier	` RoundedRectangle`	35	63	0	0
Chat-template leakage	`Assistant`, `You are a helpful...`	20	29	0	0
Punctuation only	`$`, `.`, `` ` ``	6	9	0	0
Answer without reasoning	`4`, `2`, `\boxed{326.5}`	6	23	2	8
Prompt regurgitation	problem statement restated, then EOS	3	5	0	0
Substantive attempt	a real derivation, right or wrong	11	25	4	16
Total		216	500	6	24

while a handful of degenerate strings sit at the top of the individual-probability ranking (-8.0 here). Global tempering at $\alpha = 4$ moves essentially all mass onto the latter.

K.2 What the Sharpened Samplers Actually Return

Table 6 groups the completions returned by the two sharpened samplers for which we have per-problem outputs across all of MATH500: the 216 proposals accepted by the rejection sampler over its 165 qualifying problems, and the 500 Metropolis–Hastings completions, one per problem. The median accepted proposal is 2 tokens long and the median Metropolis–Hastings completion is 1 token; $198/216$ and $461/500$ respectively are at most five tokens. For comparison, the base model's mean completion length on this benchmark is in the hundreds of tokens.

Two features of Table 6 are worth emphasizing. First, the overwhelming majority of returned completions—$199/216$ for rejection sampling and $452/500$ for Metropolis–Hastings—are not attempts at the problem at all. Second, of the completions that are graded correct, a third of the Metropolis–Hastings ones (8 of 24) and a third of the rejection-sampling ones (2 of 6) contain no reasoning: they are bare answers one or two tokens long. Reported accuracy under strong sharpening is therefore partly an artifact of answer extraction rather than a measure of problem solving.

K.3 Instructive Individual Cases

The following four cases are worth showing individually because each one complicates the simple reading that sharpening merely lowers accuracy.

A degenerate completion that is graded correct. On MATH500 problem 77, "If $2^8 = 4^x$, what is the value of x?" (reference answer 4), Metropolis–Hastings returns the single token `4`. Math-Verify extracts 4 and marks the completion correct, even though the sampler emitted no reasoning whatsoever and stopped after one token. Eight of the 24 correct Metropolis–Hastings completions on MATH500 are of this form, three of them bare tokens of length one or two (`2`, `4`, `25`) and the rest short answer statements (`\boxed{326.5}`, `Answer: \boxed{3}`, `0.35625`). Benchmark accuracy therefore does not cleanly separate solving a problem from emitting a plausible token.

An answer-only completion that is graded incorrect. On problem 37 the rejection sampler accepts the 12-token completion `The final answer is \boxed{720}.` Math-Verify extracts 720, which is wrong. This is the same surface form as the previous case with the opposite outcome: whether an unreasoned guess scores depends only on whether the guessed token happens to match.

Prompt regurgitation. On problem 179 the rejection sampler accepts a 120-token completion that begins `user\nFind the equation of the plane containing the points $(0,-1,-1),$...` and simply restates the problem before stopping. Problems 194 and 74 ac-

cept similar restatements. These completions are high probability precisely because they copy the prompt, and they show that the degeneracy is not limited to single-token artifacts.

A genuine short solution. On problem 90, "To divide 10 by $\frac{2}{3}$, what should Remmy multiply 10 by?", the rejection sampler accepts a 67-token completion that is both correct and a real derivation:

> To divide 10 by $\frac{2}{3}$, Remmy should multiply 10 by the reciprocal of $\frac{2}{3}$, which is $\frac{3}{2}$. Therefore, Remmy should multiply 10 by $\boxed{\frac{3}{2}}$.

Six of the 216 accepted proposals are correct, and the longest Metropolis–Hastings completion on MATH500 is an 810-token correct solution to a vector-projection problem. Sharpening does not make correct output impossible; on the problems where it survives, it is markedly more concise than the base model's average completion.