

# Parallel Test-Time Scaling for Latent Reasoning Models

Runyang You<sup>1</sup>, Yongqi Li<sup>1†</sup>, Meng Liu<sup>2</sup>, Wenjie Wang<sup>3</sup>, Liqiang Nie<sup>4</sup>, Wenjie Li<sup>1</sup>

<sup>1</sup>The Hong Kong Polytechnic University <sup>2</sup>Shandong Jianzhu University

<sup>3</sup>University of Science and Technology of China

<sup>4</sup>Harbin Institute of Technology (Shenzhen)

runyang.y@outlook.com liyongqi@gmail.com

## Abstract

Parallel test-time scaling (TTS) is a pivotal approach for enhancing large language models (LLMs), typically by sampling multiple token-based chains-of-thought in parallel and aggregating outcomes through voting or search. Recent advances in latent reasoning, where intermediate reasoning unfolds in continuous vector spaces, offer a more efficient alternative to explicit Chain-of-Thought, yet whether such latent models can similarly benefit from parallel TTS remains open, mainly due to the absence of sampling mechanisms in continuous space, and the lack of probabilistic signals for advanced trajectory aggregation. This work enables parallel TTS for latent reasoning models by addressing the above issues. For sampling, we introduce two uncertainty-inspired stochastic strategies: Monte Carlo Dropout and Additive Gaussian Noise. For aggregation, we design a Latent Reward Model (LatentRM) trained with step-wise contrastive objective to score and guide latent reasoning. Extensive experiments and visualization analyses show that both sampling strategies scale effectively with compute and exhibit distinct exploration dynamics, while LatentRM enables effective trajectory selection. Together, our explorations open a new direction for scalable inference in continuous spaces.

## 1 Introduction

Large language models (LLMs) have achieved remarkable performance on challenging tasks through test-time scaling (TTS), where more inference compute leads to better results. Such scaling is often realized through explicit Chain-of-Thought (CoT) reasoning (Wei et al., 2022), where models verbalize intermediate solving steps in natural language, generate long token sequences over which

compute can be scaled. One promising direction is to scale in parallel (Fu et al., 2025; Qu et al., 2025; Snell et al., 2024), which samples multiple reasoning paths and aggregates outcomes via methods such as majority voting (Wang et al., 2022), best-of- $N$  (Zhou et al., 2022), or guided search (Uesato et al., 2022). Through parallel scaling, models transform extra inference compute directly into stronger capabilities, without any need for parameter updates or retraining.

Moving beyond explicit token-by-token reasoning, recent works show that reasoning can instead unfold in latent space, with vector representations replacing tokens as the intermediate steps in autoregressive generation (Li et al., 2025; Chen et al., 2025a). This paradigm, also known as continuous CoT (CCOT), has the potential to match or even surpass explicit CoT while being more compact and computationally efficient, resembling intuitive human reasoning (Hao et al., 2024; Shen et al., 2025; Tan et al., 2025; Wu et al., 2025a). By effectively compressing or distilling explicit reasoning into continuous representations, these methods not only speed up inference but also capture abstract patterns difficult to express in natural language, positioning latent reasoning as a promising direction for advancing LLM capabilities.

Given the promising potential of latent reasoning, and that parallel TTS can effectively utilize additional generated tokens to deliver superior performance (Wang et al., 2022; Muennighoff et al., 2025; Snell et al., 2024), a natural question arises: **can latent reasoning models also benefit from parallel TTS?** Extending parallel TTS into latent reasoning models is appealing but non-trivial. First, latent reasoning models lack the fundamental sampling capability: while token-based models generate logits that enable commonly-used sampling strategies like top-k or nucleus sampling—as illustrated in Figure 1a, latent models operate on continuous vectors without an explicit probability dis-

<sup>†</sup>Corresponding author.

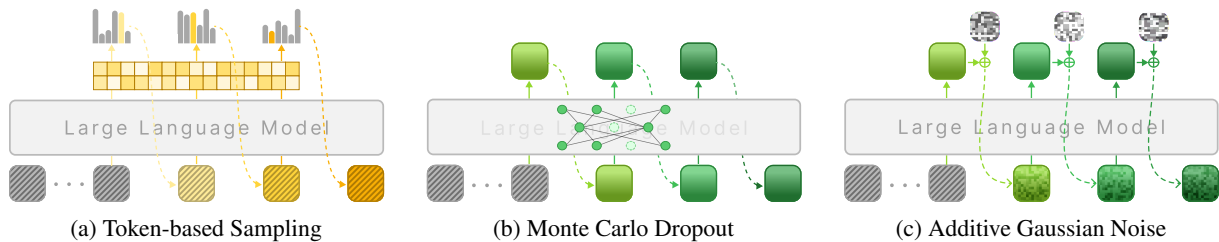


Figure 1: Sampling mechanisms for token-based generation (1a) and our proposed approaches for the latent setting (1b & 1c). (1a): Multinomial sampling over token probabilities at each step. (1b): Monte Carlo Dropout (MC-dropout): Stochastic inference via randomly sampled dropout masks. (1c): Additive Gaussian Noise (AGN): independent Gaussian perturbations injected to each latent thought.

tribution and therefore lack an inherent mechanism to stochastically generate multiple reasoning paths.

Second, the aggregation mechanism used in token-based models does not directly apply to latent reasoning models: while token-based methods typically utilize token-level probabilities to rank reasoning trajectories, latent reasoning models provide no inherent likelihoods or stepwise scores, making it difficult to evaluate or aggregate the sampled latent reasoning paths.

To address these challenges, we re-think both sampling and aggregation for latent reasoning, proposing effective solutions tailored to the continuous setting. For sampling, to ensure informativeness and controllability of the sampling space, we draw on uncertainty estimation theory (Gawlikowski et al., 2023; Liu et al., 2019) and propose two simple yet effective strategies to introduce stochasticity into latent reasoning: Monte Carlo dropout (MC-dropout) to capture epistemic uncertainty; and Additive Gaussian Noise (AGN) to simulate aleatoric uncertainty, as illustrated in Figure 1. For aggregation, we propose the Latent Reward Model (LatentRM), a dedicated scorer that evaluates and guides the progression of latent reasoning at each intermediate step. LatentRM is trained via a step-wise contrastive objective that discriminates among candidate thoughts at each reasoning step, enabling fine-grained, position-sensitive scoring.

Building on these designs, extensive experiments and visualization analyses reveal that both proposed sampling strategies not only scale effectively with increased compute but also exhibit distinct exploration dynamics in latent space: MC-Dropout promotes structured, directed expansion toward unconventional solutions, resulting in higher coverage, whereas Additive Gaussian Noise drives broad and isotropic exploration that enriches diversity around the deterministic center. For aggregation,

LatentRM enables consistent gains with best-of- $N$  and beam search across compute budgets. Ablation studies confirm that contrastive supervision and stochastic rollouts are both crucial. Overall, our findings demonstrate that parallel TTS can transfer to latent reasoning models through redesigned sampling and aggregation, opening a new pathway for scalable inference in latent space.

Collectively, core contributions are as follows:

- We introduce parallel test-time scaling into latent reasoning models, enabling a key scaling capability that was previously exclusive to token-based reasoning paradigms.
- We address the fundamental challenge of sampling in continuous latent space by proposing two complementary strategies—Monte Carlo Dropout and Additive Gaussian Noise to enable controlled and informative stochastic latent reasoning.
- We design the Latent Reward Model (LatentRM), a dedicated scorer under step-wise contrastive supervision to evaluate and guide latent reasoning, enabling effective aggregation in the latent setting.

## 2 Related Work

### 2.1 Test-Time Scaling

Large language models increasingly rely on test-time scaling (TTS)—allocating more computation at inference to improve reasoning quality. Two main axes have emerged. The sequential axis advances by generating longer reasoning traces (Muennighoff et al., 2025; Wang et al., 2025), a strategy employed in recent reasoning-oriented models (DeepSeek-AI et al., 2024; Yang et al., 2025; You et al., 2025; Fu et al., 2025; Lee et al.,

2025). The parallel axis generates multiple reasoning trajectories and aggregates them, as in Self-Consistency (Wang et al., 2022), Best-of-N (Zhou et al., 2022), and tree-structured exploration methods (Uesato et al., 2022). These two axes are often combined, enabling flexible accuracy–compute trade-offs under deployment constraints. Recent works have sought to improve the efficiency of TTS primarily by refining aggregation strategies and decision rules, such as adaptive voting and confidence-based stopping criteria (Snell et al., 2024; Brown et al., 2024). Yet all of these approaches ultimately rest on token-level sampling, decoding by drawing tokens from the predictive distribution. Such mechanism is intrinsic to token-based reasoning, but is fundamentally incompatible with latent reasoning, which operates over continuous representations.

## 2.2 Latent Reasoning

Chain-of-Thought (CoT) reasoning has proven highly effective, yet it remains constrained by its reliance on natural language (Zhu et al., 2025). This introduces inefficiency—most tokens add little value and fail to capture cognition like abstract insights or intuitive leaps, making step-by-step verbalization an unnatural limit. These limitations motivate the shift toward latent reasoning (Chen et al., 2025c,a). One line of work pursues architectural modifications, such as dynamically skipping or repeating transformer layers, or introducing auxiliary modules to adjust computational cost (Chen et al., 2025b; Geiping et al., 2025; Mohtashami et al., 2024). More recently, research has shifted to latent autoregressive reasoning, where models build trajectories directly in hidden space. COCONUT (Hao et al., 2024) pioneered latent generation using the last hidden state as the next thought, training via curriculum learning, while CODI (Shen et al., 2025) introduced latent auto-regression through self-distillation. While CoLaR (Tan et al., 2025) and SoftCoT++ (Xu et al., 2025b) made initial attempts to inject noise for stochastic latent reasoning, these efforts remain preliminary.

## 3 Preliminaries

We begin by formalizing latent reasoning and outlining the inference framework that underlies our test-time scaling approach.

**Latent Reasoning.** A latent reasoning model performs autoregressive generation in continuous hidden space. At each step, it produces a latent vector—the last hidden state from the transformer backbone—that represents an intermediate reasoning step. This formulation bypasses token-level verbalization, providing a more compact and efficient alternative to standard explicit CoT reasoning.

**Inference Process.** Let the input sequence be  $x = [x_1, x_2, \dots]$ , the latent reasoning trajectory as  $\mathbf{h}_{1:T} = [\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_T] \in \mathbb{R}^{T \times d}$ , where each  $\mathbf{h}_t \in \mathbb{R}^d$  represents the hidden state (thought) at step  $t$ . At inference time, the model generates the next latent thought autoregressively:

$$\mathbf{h}_{t+1} = f_{\theta}(\mathbf{h}_{1:t}, x),$$

where  $f_{\theta}$  is the LLM transformer blocks parameterized by  $\theta$ . This process ends with the end-of-thinking token ( $\langle |EoT| \rangle$ ), which triggers the transition to explicit token generation for the final answer.

**Reasoning Length Control.** Different latent reasoning models control the reasoning length in distinct ways. COCONUT (Hao et al., 2024) and CODI (Shen et al., 2025) use predetermined sequence lengths, where a fixed  $T$  is set and the  $\langle |EoT| \rangle$  is manually inserted. CoLaR (Tan et al., 2025) introduces dynamic latent compression: the reasoning length (or “thinking speed”) is adjusted by a compression factor specified in the prompt (e.g., “thinking speed = 5×”), allowing the model to adaptively decide when to emit the EoT token and terminate reasoning.

## 4 Method

Building on the latent reasoning formulation above, we now introduce our test-time scaling framework, which consists of two key components: stochastic sampling and latent trajectory aggregation.

### 4.1 Stochastic Sampling in Latent Space

To scale latent reasoning at test time, stochasticity must be introduced into the latent generation process, enabling the model to produce a diverse set of trajectories  $\{\mathbf{h}^{(n)}\}_{n=1}^N$ , where each  $\mathbf{h}^{(n)} = [\mathbf{h}_1^{(n)}, \dots, \mathbf{h}_T^{(n)}]$  represents a sampled sequence of latent thoughts. However, arbitrary noise can easily distort reasoning process. To ensure that sampling is both controllable and meaningful, we

draw on uncertainty estimation theory, which delineates two sources of uncertainty that provide principled probabilistic spaces for sampling: (1) *epistemic uncertainty*, reflecting variability due to the model’s limited knowledge, and (2) *aleatoric uncertainty*, arising from noise or ambiguity inherent in the inputs. Accordingly, we propose two complementary strategies: **Monte Carlo dropout** (MC-dropout) to capture epistemic uncertainty, and **Additive Gaussian Noise** (AGN) to simulate aleatoric uncertainty. An overview of the two mechanisms is shown in Figure 1b and Figure 1c.

We now detail the sampling procedures. Formal derivations and algorithmic details are provided in Appendix B and Appendix H, respectively.

**MC-dropout.** We keep dropout active at inference with rate  $p$ , sampling binary masks  $m^{(n)} \sim \text{Bernoulli}(p)$  applied to model weights  $\theta^{(n)}$ :

$$\mathbf{h}_{t+1}^{(n)} = f_{\theta^{(n)}}(\mathbf{h}_{1:t}^{(n)}, \mathbf{x}).$$

Dropout is applied after the feed-forward layer in each Transformer block (Gao et al., 2021), capturing *epistemic uncertainty* as each pass samples a different weight configuration.

**AGN.** As a complementary sampling strategy, we add isotropic Gaussian noise directly to the latent thoughts to induce controlled stochasticity, as illustrated in Figure 1c. Specifically, a random perturbation is drawn and applied at each reasoning step as

$$\epsilon_t^{(n)} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}), \quad \mathbf{h}_t^{(n)*} = \mathbf{h}_t^{(n)} + \epsilon_t^{(n)},$$

where  $\mathbf{I}$  is the identity matrix and  $\epsilon_t^{(n)}$  denotes zero-mean Gaussian noise with standard deviation  $\sigma$ . The model then continues autoregressive inference based on the perturbed trajectory:

$$\mathbf{h}_{t+1}^{(n)} = f_{\theta}(\mathbf{h}_{1:t}^{(n)*}, \mathbf{x}).$$

This procedure models *aleatoric uncertainty*, as the variance is controlled solely by the noise scale  $\sigma$ , independent of the model parameters.

## 4.2 Latent Trajectories Aggregation

A key challenge in extending parallel TTS to latent reasoning lies in aggregation. Unlike token-based TTS, where log-likelihoods provide a natural scoring mechanism, latent trajectories are continuous vectors without explicit scores. This prevents

direct application of best-of- $N$  or beam search. While process reward models (PRMs) (Zhang et al., 2024; Wang et al., 2024) could be a natural choice for evaluating intermediate reasoning steps, latent thoughts are abstract vectors without linguistic form and cannot be interpreted or scored by existing PRMs. To this end, we propose the **Latent Reward Model (LatentRM)**, a dedicated reward model for latent thoughts that enables effective aggregation strategies.

**Architecture.** We extend the latent reasoning backbone with an additional scoring head that maps hidden states to a scalar. LatentRM  $g_{\phi}$  takes in the input prompt  $x$  and the generated latent trajectory up to step  $t$  and outputs a score:

$$r_t = g_{\phi}(\mathbf{x}, \mathbf{h}_{1:t}) \in \mathbb{R},$$

which estimates the promise of continuing from the current thought  $\mathbf{h}_t$ .

**Inference.** During inference, LatentRM evaluates each candidate trajectory by computing the sum of logits  $\sum_t r_t$  over the generated sequence, serving as a proxy for the relative quality of thought  $\mathbf{h}_{1:t}$ . This logit-summing strategy is justified in Appendix C, which shows that cumulative logits can solely determine trajectory ranking.

**Data.** To supervise the learning of LatentRM, we construct thought-level quality labels by estimating the empirical correctness of each intermediate thought. Specifically, for each input  $x$ , we sample  $N$  trajectories  $\mathbf{H} = \{\mathbf{h}^{(n)}\}_{n=1}^N$ , and for every thought within every trajectory, we rollout  $M$  stochastic completions, obtaining a set of final answers  $\{a_m\}_{m=1}^M$ , and compute

$$\tilde{y} = \frac{1}{M} \sum_{m=1}^M \mathbb{I}\{a_m = a^*\},$$

as a proxy for the quality of thought, where  $a^*$  denotes the ground-truth answer and  $\mathbb{I}\{\cdot\}$  as the indicator function.

**Objective.** A straightforward approach is to treat each thought independently and optimize LatentRM with binary cross-entropy (BCE) loss, which is commonly used for training PRMs. However, this approach performs poorly in practice, as it provides only isolated supervision per candidate and lacks relative comparison across thoughts at the same step. To address this, we adopt a step-wise

contrastive formulation. At each step  $t$ , the scores of all  $N$  candidates are compared via a softmax,

$$p_t^{(n)} = \frac{\exp(r_t^{(n)})}{\sum_{n'=1}^N \exp(r_t^{(n')})}.$$

LatentRM is then trained with the negative log-likelihood loss

$$\mathcal{L} = - \sum_t \sum_{n=1}^N y_t^{(n)} \log p_t^{(n)}.$$

## 5 Experiment Setup

### 5.1 Benchmarks and Models.

**Benchmarks.** Following previous works (Shen et al., 2025; Tan et al., 2025; Hao et al., 2024), evaluation is conducted on three benchmarks: (1) GSM8K-Test, the official test split of GSM8K (1,319 test samples)<sup>1</sup>; (2) GSM8K-Hard (Gao et al., 2022), a more challenging version of GSM8K-Test where numbers are scaled to larger magnitudes to increase problem difficulty with (1,319 test samples)<sup>2</sup>, and (3) MultiArith (Uesato et al., 2022), a dataset focusing on multi-step arithmetic reasoning (600 test samples)<sup>3</sup>.

**Latent reasoning models/backbones.** We evaluate five representative latent reasoning models: (1) COCONUT (Hao et al., 2024), which progressively replaces CoT steps with latent thoughts; (2) CODI (Shen et al., 2025), which self-distills CoT into latent space; and (3) CoLaR (Tan et al., 2025), which performs dynamic latent compression with reinforcement learning. All experiments leverage officially released checkpoints, where COCONUT and CODI are backboneed on GPT-2 (124M), and CoLaR on Llama-3.2-1B. (4) Latent-SFT (Llama-3.2-1B-Instruct) (Deng et al., 2025) and (5) Render-of-Thought (RoT) with Qwen3-VL at 2B and 4B parameter scales (Wang et al., 2026). Following the default configurations in the original papers, for COCONUT and CODI, we fix the number of latent thoughts to  $T = 6$ ; for CoLaR, thinking speed set to 2. For ColaR, Latent-SFT and RoT, we set the maximum number of latent thoughts to 64.

<sup>1</sup><https://huggingface.co/datasets/gsm8k>

<sup>2</sup><https://huggingface.co/datasets/reasoning-machines/gsm-hard>

<sup>3</sup>[https://huggingface.co/datasets/lighteval/multi\\_arith](https://huggingface.co/datasets/lighteval/multi_arith)

### 5.2 Sampling Evaluation

To evaluate the effectiveness of different stochastic sampling methods, we measure how well each method scales with the number of sampled trajectories. Specifically, *coverage* quantifies the fraction of problems for which at least one of the  $N$  sampled trajectories yields the correct answer:

$$\text{coverage} = \frac{1}{|D|} \sum_{x \in D} \mathbb{I}\{\exists n \leq N : a^{(n)} = a^*\},$$

This metric is equivalent to *pass@k* when sampling size  $N$  and cutoff  $k$  are identical (Zhang et al., 2024; Brown et al., 2024). Higher coverage indicates stronger sampling effectiveness—i.e., a greater ability to uncover correct reasoning paths as the inference budget increases. All sampling schemes are evaluated under equal  $N$  for fair comparison.

### 5.3 Aggregation Evaluation

We evaluate the proposed LatentRM through two of its supported aggregation strategies: (1) *best-of- $N$*  selection scored by LatentRM, (2) *beam search* guided by LatentRM, compared against *majority voting* as a non-parametric baseline. We adopt MC-dropout with  $p = 0.2$  as the default sampling setting. To ensure fairness, we match all methods under the same compute budget: *best-of- $N$*  and *majority voting* use  $N$  independent samples, while *beam search* adopts a beam size of  $\sqrt{N}$  for comparable decoding cost (Zhang et al., 2024; Snell et al., 2024). Training configurations and optimization details for LatentRM are provided in Appendix G, and full inference procedures for aggregation strategies are described in Appendix D.

## 6 Results of Sampling

### 6.1 Main Results

We systematically evaluate both stochastic sampling strategies by varying the sample count ( $N$ ) and measuring solution coverage. We tune the AGN hyperparameter  $\sigma$  over  $[0.01, 1.5]$  and the MC-Dropout probability  $p$  over  $[0, 1]$ , using binary search within each interval to maximize coverage@64. We then plot two curves using the optimal hyperparameters for each method. Additional results on harder benchmarks are provided in Appendix E. For deployment without exhaustive sweeps, we use the following **heuristic ranges** informed by backbone architecture: (i) GPT-2-style

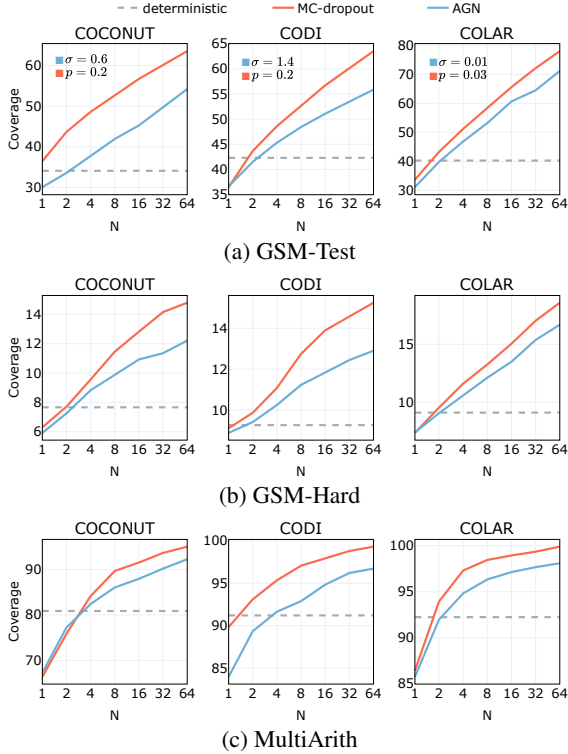


Figure 2: Coverage (%) versus  $N$  plot for COCONUT, CODI, and CoLaR on GSM-Test (2a), GSM-Hard (2b) and MultiArith (2c). Each subplot compares MC-dropout and AGN using the optimal hyperparameter. Higher coverage indicates a larger fraction of problems solved by  $N$  attempts. Results are reported as the mean over three runs.

models (e.g., COCONUT, CODI) work well with dropout rate  $p \in [0.1, 0.3]$  and noise scale  $\sigma \in [0.5, 0.7]$ ; (ii) Llama-3.2-1B-style models (e.g., CoLaR, Latent-SFT) are more stable with smaller perturbations,  $p \in [0.01, 0.03]$  and  $\sigma \in [0.01, 0.03]$ . In practice, fixing  $(p, \sigma)$  to a single pair within these bands and using it across benchmarks yields coverage within a negligible gap of values obtained by per-dataset tuning, so practitioners need not re-tune for each split. Figure 2 presents four key findings: (1) both MC-dropout and AGN can effectively scale, as coverage increases monotonically with larger sample sizes; (2) the marginal improvement diminishes as  $N$  grows, suggesting a saturation effect where additional samples contribute less to coverage gains; (3) increased sampling narrows inter-model performance gaps. Notably, at  $N=64$ , COCONUT and CODI achieve nearly equivalent coverage, despite CODI’s clear superiority at  $N=1$ . (4) MC-dropout consistently achieves higher coverage across nearly all  $N$ , highlighting its advantage as a more reliable stochastic sampling approach.

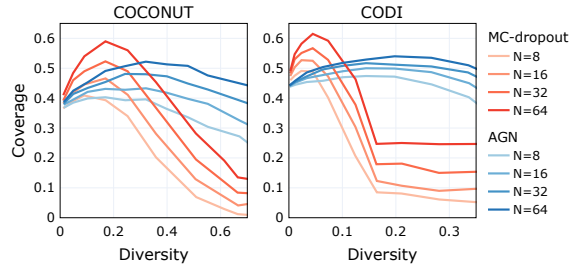


Figure 3: Coverage versus diversity for MC-dropout (red) and AGN (blue) with  $N \in \{4, 8, 16, 32\}$  by sweeping  $p$  and  $\sigma$  to span a range of diversity values. Darker shades indicate larger  $N$ . Results are shown for COCONUT (left) and CODI (right) on GSM-Test.

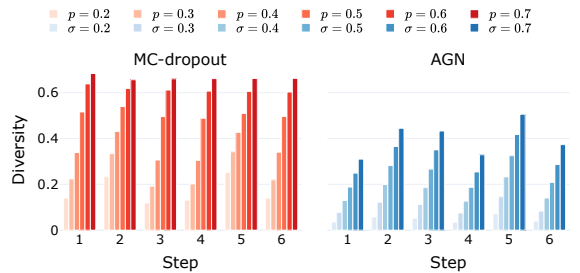


Figure 4: Diversity of latent trajectories across reasoning steps on GSM-Test with COCONUT. Left: MC-dropout ( $p = 0.1 - 0.5$ ). Right: AGN ( $\sigma = 0.1 - 0.5$ ).

## 6.2 Generalization to Larger Backbones

To evaluate the architectural robustness and scalability of our framework, we extend our experiments to more capable backbones, including Latent-SFT (Deng et al., 2025) and Render-of-Thought (RoT) (Wang et al., 2026), with parameter counts up to 4B. As shown in Table 1, our parallel TTS framework yields consistent performance improvements across various architectures and scales. In particular, on the challenging MATH500 benchmark, RoT-4B improves from 20.3% to 22.0% with  $N = 16$ , highlighting the framework’s ability to generalize to larger models and complex mathematical reasoning tasks.

## 6.3 Analysis on Diversity

The key question to ask is: *what makes a good sampling strategy for latent reasoning?* Beyond simply scaling the sampling budget  $N$  to boost problem-solving success, a good sampling strategy should encourage diverse reasoning trajectories rather than expending computational effort on redundant or overly similar paths. Importantly, this pursuit of diversity must not come at the cost of overall coverage.

Backbone	Benchmark	Det.	Cov@8	Cov@16
<b>Latent-SFT</b> (Llama-3.2-1B)	GSM8K	0.445	0.585	0.649
	MultiArith	0.934	0.962	0.967
<b>RoT-4B</b> (Qwen3-VL)	GSM8K	0.375	0.394	0.397
	GSM8K-Hard	0.139	0.143	0.143
	MATH500	0.203	0.218	0.220
<b>RoT-2B</b> (Qwen3-VL)	GSM8K	0.233	0.257	0.257
	GSM8K-Hard	0.086	0.091	0.092
	MATH500	0.115	0.128	0.130

Table 1: Generalization across larger backbones and scales.  $N$  denotes the number of sampled latent trajectories. MC-dropout is adopted as the default setting.

**Definition.** To formalize this intuition, we introduce the notion of sampling *diversity* as the average pairwise cosine dissimilarity among latent thoughts across reasoning steps:

$$\text{diversity} = \frac{1}{|D|T} \sum_{x \in D} \sum_{t=1}^T d_t(x),$$

where  $d_t(x) = \frac{2}{N(N-1)} \sum_{i < j} (1 - \cos(\mathbf{h}_t^{(i)}, \mathbf{h}_t^{(j)}))$ , measures the dissimilarity at step  $t$ . A higher diversity score reflects greater variability in the reasoning paths taken, suggesting a richer exploration of the problem space.

**Diversity vs. coverage.** To understand the interplay between diversity and performance, we plot coverage against diversity across different sampling method, while sweeping  $p$  and  $\sigma$  under varying  $N$ . The resulting trade-off curves are visualized in Figure 3. Key insights are as follow. (1) Across models and methods, a clear “sweet spot” emerges, with coverage peaking at moderate diversity. This suggests that while some level of stochasticity is beneficial, excessive or insufficient diversity can hinder performance; and (2) at larger diversity levels, AGN tends to maintain or even improve coverage, whereas MC-dropout shows a sharp decline. This highlights AGN’s superior ability to preserve solution quality even when the injected randomness is high, making it a more robust choice for high-diversity exploration. Overall, if high-diversity exploration is desired, AGN serves as a more reliable choice.

**Step-wise dynamics.** To examine how sampling affects reasoning over time, we analyze diversity at each step. As shown in Figure 4, MC-dropout maintains similar diversity across steps for a given  $p$ , reflecting its adaptivity: stochasticity follows

the model’s own uncertainty, enabling consistent exploration. In contrast, AGN shows pronounced fluctuations because a fixed  $\sigma$  perturbs latent vectors of varying scales unevenly, yielding inconsistent stochastic influence. These patterns match each method’s mechanism: MC-dropout modulates noise by epistemic uncertainty, supporting effective exploration, whereas AGN, though diversity can be increased via  $\sigma$ , injects less adaptive noise with variable impact across steps.

## 6.4 Visualization on Stochastic Latent Landscape

To unveil how stochasticity reshapes the hidden landscape of reasoning, we cast sampled latent thoughts into a 2D stage via t-SNE (Figure 5). What emerges is strikingly distinct: Dropout produces a *directional drift*—dense and contiguous along specific directions, whereas AGN yields an *isotropic radial dispersion*, a “firework” pattern with broader area but lower local density.

These geometric signatures explain their different behaviors on easy and hard questions. For easy cases (Figure 5a), where correct regions lie near the deterministic latent, AGN maintains accuracy by keeping more probability mass around the center, whereas large  $p$  dropout drifts away and degrades performance. For hard cases (Figure 5b), where correct regions are farther away, dropout’s larger displacement and denser exploration increase the chance of reaching the correct solution. Ultimately, the geometry of exploration explains their complementary strengths: MC-dropout excels on harder tasks, whereas AGN maintains robustness even under strong stochasticity.

## 7 Results of Aggregation

### 7.1 Main Results

We report the TTS results using the three above-mentioned aggregation methods in Figure 6. First, accuracy increases monotonically with  $N$  across all three datasets, demonstrating that latent reasoning can effectively scale with more inference compute. Second, both Best-of- $N$  and Beam Search consistently outperform Majority Voting, confirming that LatentRM can effectively distinguish promising reasoning trajectories. Third, Beam Search performs comparably with Best-of- $N$  on GSM-Test and MultiArith but trails on GSM-Hard, suggesting that early-step score noise can cause premature pruning in more challenging problems. Finally,

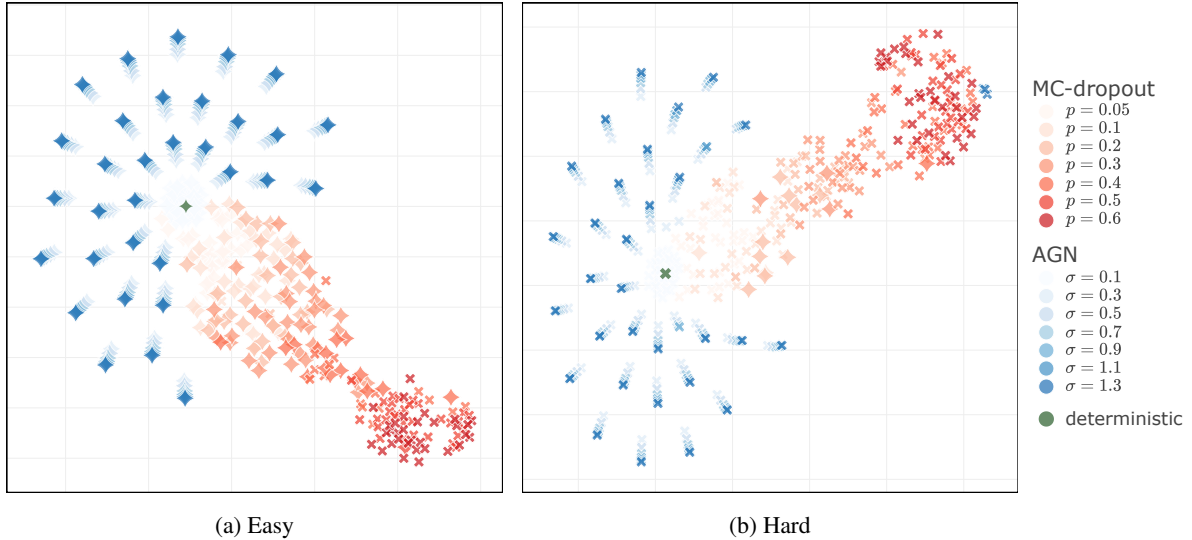


Figure 5: t-SNE visualization of latent thoughts sampled with different dropout rates ( $p$  from light to dark) and Gaussian-noise scales ( $\sigma$  from light to dark). The green marker denotes the deterministic latent thought (no stochasticity). Diamonds ( $\diamond$ ) indicate correct reasoning trajectories; crosses ( $\times$ ) indicate incorrect ones. (5a): an easy question. (5b): a hard question.

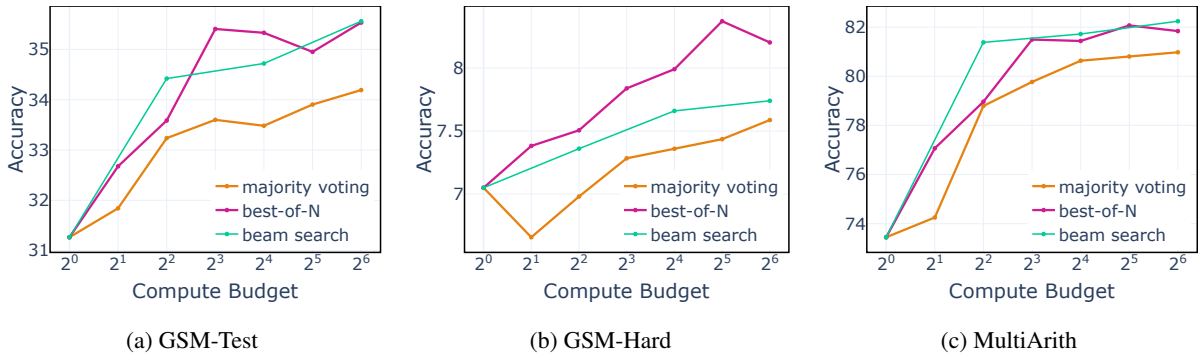


Figure 6: Test-time scaling results with majority voting, best-of- $N$ , and beam search for latent reasoning. Accuracy(%) versus compute budget  $N$  under three aggregation strategies on (6a) GSM8K-Test, (6b) GSM8K-Hard, and (6c) MultiArith. Results are reported as the mean over three runs.

the gains are most pronounced on MultiArith, highlighting the generalization ability of the learned scorer across arithmetic reasoning patterns. Overall, LatentRM enables scalable and reliable aggregation, with Best-of- $N$  emerging as the most effective route for latent test-time scaling.

## 7.2 Ablation Studies

To understand the contribution of each component in LatentRM, we conduct ablation studies under the Best-of- $N$  setting with  $N = 8$  on GSM-Test and GSM-Hard (Table 2). Each variant disables one design element of LatentRM while keeping all other configurations identical. (1) Removing the step-wise contrastive loss (*w/o contrastive*) causes a noticeable drop, showing that relative supervision among concurrent thoughts provides stronger

Variant	Test	Hard
Best-of-8 with LatentRM	<b>35.4</b>	<b>7.8</b>
w/o contrastive (BCE)	33.5	7.4
w/o stochastic rollouts	30.7	6.0
random scalar head (untrained)	28.9	5.8
Majority Voting	33.6	6.1

Table 2: **Ablation studies of LatentRM.** We report accuracy (%) on GSM-Test and GSM-Hard under Best-of- $N$  aggregation ( $N = 32$ ). Each variant isolates one design choice in LatentRM.

learning signals than isolated binary labels. (2) Excluding stochastic rollouts (*w/o stochastic rollouts*), where each thought is labeled only by the final trajectory correctness instead of intermediate estimates, leads to a further decline, highlighting the

importance of Monte Carlo estimation for reliable thought-level annotation. (3) Using an untrained random scalar head (*random head*) performs even worse—below Majority Voting—indicating that the gain stems from learned evaluation rather than architectural modification. Together, these findings verify that LatentRM effectively learns to assess latent trajectories and is crucial for successful aggregation in latent test-time scaling.

### 7.3 More Analysis

Due to space constraints, further extended analyses on *Comparison of Latent and Explicit Reasoning with TTS*, *Wall-Clock Comparison of Latent and Explicit Reasoning with TTS*, and *Analysis on LatentRM under Variable-Length Setting* are provided in Appendix F.1, Appendix F.2, and Appendix F.3.

## 8 Conclusion and Future Work

This paper proposed a parallel test-time scaling framework for latent reasoning models, addressing the central challenges of sampling in continuous latent spaces and aggregating latent trajectories. By introducing two principled stochastic sampling methods—Monte Carlo Dropout and additive Gaussian noise—rooted in uncertainty theory, and developing a latent Reward Model trained with step-wise contrastive objective for effective trajectory aggregation, our approach enables scalable and robust parallel inference in the latent regime. Extensive experiments and analyses demonstrate distinctive exploration behaviors, robust aggregation through latentRM, and consistent performance scaling across compute budgets, yielding new insights into test-time scaling for latent reasoning.

Building on this foundation, future research should explore two key directions: (1) integrating sampling and aggregation into a reinforcement learning framework to optimize latent trajectories through iterative feedback and reward shaping; and (2) investigating how to adapt ensemble techniques (Zhao et al., 2024; Sun et al., 2024; Xu et al., 2025a) to latent reasoning, addressing the key challenge of applying token-based ensemble methods to continuous latent representations.

### Acknowledgment

The work described in this paper was supported by the Research Grants Council of Hong Kong (PolyU/15207122, PolyU/15213323, PolyU/15209724, PolyU/15205325), the PolyU

internal grants (BDWP), and the Special Fund for Taishan Scholar Project of Shandong Province.

### Limitation

This work provides an effective framework for parallel test-time scaling in latent reasoning models, but there are certain limitations: (1) Engineering considerations for real-time deployment. The framework shows consistent performance and generality across benchmarks and compute budgets, but real-time use may require additional engineering optimization. Nevertheless, as the core contribution is a proof-of-concept and in-depth analysis of parallel scaling for latent reasoning models, these engineering considerations do not impact the validity or the key insights presented in this study. (2) Hyperparameter sensitivity. Our sampling strategies demonstrate strong scaling results with controllable randomness. But optimal performance depends on tuning the dropout rate ( $p$ ) and noise scale ( $\sigma$ ). While this could be seen as an additional configuration burden, we note that the hyperparameter search process is relatively straightforward and does not require extra model training or complex optimization. The main contribution of this work remains focused on the theoretical framework and empirical performance, thus the potential overhead from hyperparameter tuning is minor compared to the overall benefits.

### Ethics Statement

All data, models and other related experimental artifacts used in this study are publicly released and accessible under permissive licenses for research purposes. The research procedures, data handling, and analyses do not involve harm to individuals, communities, or the environment, and no personal or sensitive information is included.

### References

- Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. [Large Language Monkeys: Scaling Inference Compute with Repeated Sampling](#). *arXiv preprint*.
- Xinghao Chen, Anhao Zhao, Heming Xia, Xuan Lu, Hanlin Wang, Yanjun Chen, Wei Zhang, Jian Wang, Wenjie Li, and Xiaoyu Shen. 2025a. [Reasoning Beyond Language: A Comprehensive Survey on Latent Chain-of-Thought Reasoning](#). *arXiv preprint*.

- Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. 2025b. [Inner Thinking Transformer: Leveraging Dynamic Depth Scaling to Foster Adaptive Internal Thinking](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28241–28259, Vienna, Austria. Association for Computational Linguistics.
- Yuxin Chen, Yiran Zhao, Yang Zhang, An Zhang, Kenji Kawaguchi, Shafiq Joty, Junnan Li, Tat-Seng Chua, Michael Qizhe Shieh, and Wenxuan Zhang. 2025c. [The Emergence of Abstract Thought in Large Language Models Beyond Any Language](#). *arXiv preprint*.
- DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2024. [DeepSeek-V3 Technical Report](#). *arXiv preprint*.
- Jingcheng Deng, Liang Pang, Zihao Wei, Shichen Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. 2025. [Latent reasoning in llms as a vocabulary-space superposition](#). *Preprint*, arXiv:2510.15522.
- Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. 2025. [Deep Think with Confidence](#). *arXiv preprint*.
- Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In *Proceedings of the 33rd international conference on international conference on machine learning - volume 48, ICML'16*, pages 1050–1059, New York, NY, USA. JMLR.org.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. [PAL: Program-aided language models](#). *arXiv preprint arXiv:2211.10435*.
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. [SimCSE: Simple Contrastive Learning of Sentence Embeddings](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Jakob Gawlikowski, Cedrique Rovile Njietcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. 2023. [A survey of uncertainty in deep neural networks](#). *Artificial Intelligence Review*, 56(Suppl 1):1513–1589.
- Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. 2025. [Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach](#). *arXiv preprint*.
- Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024. [Training Large Language Models to Reason in a Continuous Latent Space](#). *arXiv preprint*.
- Kuang-Huei Lee, Ian Fischer, Yueh-Hua Wu, Dave Marwood, Shumeet Baluja, Dale Schuurmans, and Xinyun Chen. 2025. [Evolving Deeper LLM Thinking](#). *arXiv preprint*.
- Jindong Li, Yali Fu, Li Fan, Jiahong Liu, Yao Shu, Chengwei Qin, Menglin Yang, Irwin King, and Rex Ying. 2025. [Implicit Reasoning in Large Language Models: A Comprehensive Survey](#). *arXiv preprint*.
- Jeremiah Liu, John Paisley, Marianthi-Anna Kioumourtzoglou, and Brent Coull. 2019. [Accurate uncertainty estimation and decomposition in ensemble learning](#). In *Advances in neural information processing systems*, volume 32. Curran Associates, Inc.
- Amirkeivan Mohtashami, Matteo Pagliardini, and Martin Jaggi. 2024. [CoTFormer: A Chain-of-Thought Driven Architecture with Budget-Adaptive Computation Cost at Inference](#). *arXiv preprint*.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. [s1: Simple test-time scaling](#). *arXiv preprint*.
- Yuxiao Qu, Matthew Y. R. Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, and Aviral Kumar. 2025. [Optimizing Test-Time Compute via Meta Reinforcement Fine-Tuning](#). *arXiv preprint*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners.
- Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025. [CODI: Compressing Chain-of-Thought into Continuous Space via Self-Distillation](#). *arXiv preprint*.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. [Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters](#). *arXiv preprint*.
- Yetong Sun and 1 others. 2024. Breaking the ceiling of the llm community by treating token generation as a classification for ensembling. In *Findings of the Association for Computational Linguistics: EMNLP 2024*.
- Wenhui Tan, Jiase Li, Jianzhong Ju, Zhenbo Luo, Jian Luan, and Ruihua Song. 2025. [Think Silently, Think Fast: Dynamic Latent Compression of LLM Reasoning Chains](#). *arXiv preprint*.

- Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. [Solving math word problems with process- and outcome-based feedback.](#) *arXiv preprint*.
- Jian Wang, Boyan Zhu, Chak Tou Leong, Yongqi Li, and Wenjie Li. 2025. [Scaling over Scaling: Exploring Test-Time Scaling Plateau in Large Reasoning Models.](#) *arXiv preprint*.
- Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. [Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations.](#) In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. [Self-Consistency Improves Chain of Thought Reasoning in Language Models.](#)
- Yifan Wang, Shiyu Li, Peiming Li, Xiaochen Yang, Yang Tang, and Zheng Wei. 2026. [Render-of-thought: Rendering textual chain-of-thought as images for visual latent reasoning.](#) *Preprint*, arXiv:2601.14750.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. [Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.](#)
- Haoyi Wu, Zhihao Teng, and Kewei Tu. 2025a. [Parallel Continuous Chain-of-Thought with Jacobi Iteration.](#) *arXiv preprint*.
- Junhong Wu, Jinliang Lu, Zixuan Ren, Gangqiang Hu, Zhi Wu, Dai Dai, and Hua Wu. 2025b. [Llms are single-threaded reasoners: Demystifying the working mechanism of soft thinking.](#) *arXiv preprint arXiv:2508.03440*.
- Yangyifan Xu, Jianghao Chen, Junhong Wu, and Jiajun Zhang. 2025a. [Hit the sweet spot! span-level ensemble for large language models.](#) In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*.
- Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025b. [SoftCoT++: Test-Time Scaling with Soft Chain-of-Thought Reasoning.](#) *arXiv preprint*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 Technical Report.](#) *arXiv preprint*.
- Runyang You, Yongqi Li, Xinyu Lin, Xin Zhang, Wenjie Wang, Wenjie Li, and Liqiang Nie. 2025.  [\$\text{R}^2\text{ec}\$ : Towards Large Recommender Models with Reasoning.](#) *arXiv preprint*.
- Kaiyan Zhang, Jiayuan Zhang, Haoxin Li, Xuekai Zhu, Ermo Hua, Xingtai Lv, Ning Ding, Biqing Qi, and Bowen Zhou. 2024. [OpenPRM: Building Open-domain Process-based Reward Models with Preference Trees.](#)
- Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. 2025. [Soft Thinking: Unlocking the Reasoning Potential of LLMs in Continuous Concept Space.](#) *arXiv preprint*.
- Yilun Zhao and 1 others. 2024. [Bridging the gap between different vocabularies for llm ensemble.](#) In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2022. [Least-to-Most Prompting Enables Complex Reasoning in Large Language Models.](#)
- Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, Tianle Cai, Taylor Kergan, Assel Kembay, Andrew Smith, Chenghua Lin, Binh Nguyen, Yuqi Pan, Yuhong Chou, Zefan Cai, and 14 others. 2025. [A Survey on Latent Reasoning.](#) *arXiv preprint*.

## A Soft and Latent Reasoning

Soft Thinking (Wu et al., 2025b; Zhang et al., 2025) and Latent Reasoning (Hao et al., 2024; Tan et al., 2025) represent two distinct paradigms for reasoning in large language models. Soft Thinking operates in the token probability space, where "soft tokens" are created by mixing token embeddings according to probability distributions over the vocabulary. This approach effectively retains the vocabulary structure while enabling continuous interpolation between discrete tokens (Zhang et al., 2025). In contrast, Latent Reasoning works directly with pure latent vectors (hidden states) that are not constrained to be mixtures of token embeddings, operating in a more abstract and unconstrained continuous space (Hao et al., 2024).

Wu et al. (2025a) reveal that Soft Thinking suffers from a greedy pitfall: LLMs predominantly rely on the highest-probability token, suppressing alternative reasoning paths. Their Stochastic Soft Thinking addresses this by introducing controlled randomness (via Gumbel-Softmax) in soft embedding composition. In contrast, our Parallel TTS for Latent Reasoning operates in pure latent vector space, unconstrained by token embeddings

or vocabulary structure. We combine latent sampling (MC-Dropout or AGN) with a Latent Reward Model to enable parallel exploration and aggregation, discovering reasoning patterns unbound by vocabulary structure.

## B Theoretical Perspectives on Sampling in Latent Space

We here briefly derive how MC-dropout can be interpreted as an approximate Bayesian estimator of *epistemic* uncertainty, while additive Gaussian noise acts as a perturbation mechanism that simulates *aleatoric* variability in latent representations.

**MC-dropout** can be interpreted as an approximate Bayesian method, where stochastic forward passes approximate the posterior predictive distribution given the training dataset  $D$  (Gal and Ghahramani, 2016). Let the predictive distribution be

$$p(y | x, D) = \int p(y | x, \omega) p(\omega | D) d\omega,$$

where  $\omega$  denotes network weights and  $D$  the training data. Since the posterior  $p(\omega | D)$  is intractable, dropout approximates it with a variational distribution

$$q(\omega) : W_i = M_i \cdot \text{diag}(z_i), \quad z_{i,j} \sim \text{Bernoulli}(p_i).$$

At inference, performing stochastic forward passes with dropout corresponds to sampling  $\omega^{(t)} \sim q(\omega)$  and evaluating

$$y^{(n)} = f(x; \omega^{(n)}).$$

The empirical mean and variance over  $N$  samples approximate the true Bayesian predictive distribution:

$$\mathbb{E}[y^* | x] \approx \frac{1}{N} \sum_{t=1}^N f(x; \omega^{(n)}),$$

$$\text{Var}[y^* | x] \approx \frac{1}{N} \sum_{t=1}^N f(x; \omega^{(n)})^2 - \left( \mathbb{E}[y^* | x] \right)^2.$$

As  $T \rightarrow \infty$ , this Monte Carlo procedure converges to the predictive distribution under the variational approximation. Applying dropout at inference effectively samples from a distribution over model weights and creating an ensemble of predictions for each input. The variability across these predictions arises from the uncertainty in model parameters due to limited knowledge gained from training data  $D$ , this is defined as *epistemic uncertainty*.

**AGN** perturbs the latent representations directly, injecting variance proportional to the model’s local sensitivity.

$$x^* = x + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I).$$

The predictive distribution then becomes

$$p(y | x^*) = \int p(y | x + \epsilon) p(\epsilon) d\epsilon,$$

with  $p(\epsilon)$  an isotropic Gaussian prior. Expanding to first order (via Taylor approximation around  $x$ ),

$$f(x + \epsilon) \approx f(x) + J_f(x)\epsilon,$$

where  $J_f(x)$  is the Jacobian of the network output w.r.t.  $x$ . Taking expectation yields

$$\mathbb{E}[f(x + \epsilon)] \approx f(x),$$

$$\text{Var}[f(x + \epsilon)] \approx J_f(x) \Sigma_\epsilon J_f(x)^\top \quad \Sigma_\epsilon = \sigma^2 I.$$

AGN injects variance proportional to the network’s local sensitivity. Since this variability is induced externally by perturbations to the latent inputs, it corresponds to *aleatoric uncertainty*, i.e., randomness arising from inherent noise in the input space rather than uncertainty about the model itself.

## C Derivation of Cumulative Scoring

Here we derive in detail why summing the logits over each candidate trajectory serves as a valid proxy for the relative quality of the generated thought sequence. Consider  $N$  reasoning trajectories, where each trajectory  $n \in 1, \dots, N$  produces a sequence of latent scores  $r_{t'}^{(n)}$   $t'=1$ .

At each step  $t'$ , the probability assigned to thought  $n$  under the softmax over all candidates is

$$p_{t'}^{(n)} = \frac{\exp(r_{t'}^{(n)})}{\sum_{n'=1}^N \exp(r_{t'}^{(n')})}.$$

The log-probability is

$$\log p_{t'}^{(n)} = r_{t'}^{(n)} - \log \sum_{n'=1}^N \exp(r_{t'}^{(n')}).$$

The cumulative log probability over the first  $t$  time steps for thought  $n$  is:

$$\begin{aligned} \log p_{1:t}^{(n)} &= \sum_{t'=1}^t \log p_{t'}^{(n)} \\ &= \sum_{t'=1}^t r_{t'}^{(n)} - \sum_{t'=1}^t \log \sum_{n'=1}^N \exp(r_{t'}^{(n')}). \end{aligned}$$

The second term  $\sum_{t'=1}^t \log \sum_{n'=1}^N \exp(r_{t'}^{(n)})$  is identical for all trajectories at step  $t$ . Therefore, when comparing trajectories, the relative ordering of  $\log p_{1:t}^{(n)}$  depends only on the first term  $\sum_{t'=1}^t r_{t'}^{(n)}$ .

## D Inference Procedures for Aggregation

We detail the inference algorithms used to aggregate multiple latent reasoning trajectories, corresponding to the three strategies evaluated in Section 7. All procedures operate on  $N$  sampled latent trajectories  $\{h_{1:T}^{(n)}\}_{n=1}^N$  obtained via the stochastic sampling methods introduced in Section 4.1. Each latent thought  $h_t^{(n)}$  is scored by the Latent Reward Model (LatentRM) as  $r_t^{(n)} = g_\phi(x, h_{1:t}^{(n)})$ .

**Best-of- $N$ .** Each trajectory is evaluated independently using the cumulative latent reward

$$R^{(n)} = \sum_{t=1}^T r_t^{(n)}.$$

The trajectory with the highest total reward is selected for final decoding:

$$h_{1:T}^* = \arg \max_n R^{(n)}.$$

The final answer token sequence  $y^*$  is then produced by the reasoning model conditioned on  $h_{1:T}^*$ :

$$y^* = f_\theta(x, h_{1:T}^*).$$

This mirrors best-of- $N$  decoding in token-based TTS, but replaces token log-likelihoods with learned latent rewards.

**Beam Search.** We further employ a beam search guided by LatentRM to explore high-reward reasoning paths. At step  $t$ , the model expands each partial latent trajectory in the beam  $\mathcal{B}_{t-1}$  by one autoregressive step, producing  $K$  candidate extensions via stochastic sampling:

$$\tilde{h}_{1:t}^{(k)} = [h_{1:t-1}^{(b)}, \hat{h}_t^{(k)}], \quad \hat{h}_t^{(k)} \sim f_\theta(h_{1:t-1}^{(b)}, x),$$

where  $b$  indexes a beam element. LatentRM assigns scores  $r_t^{(k)}$  to all candidates, and their cumulative rewards are updated:

$$R_t^{(k)} = R_{t-1}^{(b)} + r_t^{(k)}.$$

The top- $B$  candidates by cumulative reward are retained as the next beam:

$$\mathcal{B}_t = \text{TopB}(\{\tilde{h}_{1:t}^{(k)}\}, R_t^{(k)}).$$

Decoding terminates when all beams emit the end-of-thinking token or reach the maximum latent step  $T$ . The best final trajectory is selected by its cumulative reward. In experiments, we set  $B = \sqrt{N}$  to match compute cost with best-of- $N$  as discussed in Section 5.3.

**Majority Voting.** As a non-parametric baseline, each latent trajectory is independently decoded into an answer  $a^{(n)}$ . The final output is the most frequent answer among the  $N$  candidates:

$$y^* = \text{mode}(\{a^{(n)}\}_{n=1}^N).$$

This strategy disregards latent scores and serves to isolate the benefit of learned aggregation via LatentRM.

**Implementation Details.** All aggregation procedures are executed under identical compute budgets. For best-of- $N$  and majority voting,  $N$  full trajectories are sampled; for beam search, the beam size  $B = \sqrt{N}$  and per-step expansion  $K = B$  ensure comparable total forward passes. Cumulative rewards  $\sum_t r_t^{(n)}$  are pre-normalized by trajectory length to prevent bias toward longer reasoning chains.

## E Additional Results on Harder Benchmarks

We further evaluate our method on harder benchmarks, GPQA and AIME. As shown in Figure 7, our method induces consistent performance gains even on these harder benchmarks, which demonstrates the general effectiveness of the proposed approach. However, absolute performance remains limited. We attribute to two factors: (a) small model size (e.g., GPT-2-124M, Llama-3.2-1B), and (b) the inherent challenges of the "latent reasoning" framework, which is still an evolving paradigm and struggle on advanced mathematics (AIME) and Ph.D.-level benchmark (GPQA).

## F More Analysis

### F.1 Comparison of Latent and Explicit Reasoning with TTS

To investigate the synergy between latent reasoning and Test-Time Scaling (TTS), we compare NO-COT (direct answering), explicit COT, and CO-CONUT (latent reasoning), evaluating each with and without parallel TTS ( $N = 4$ ). We measure

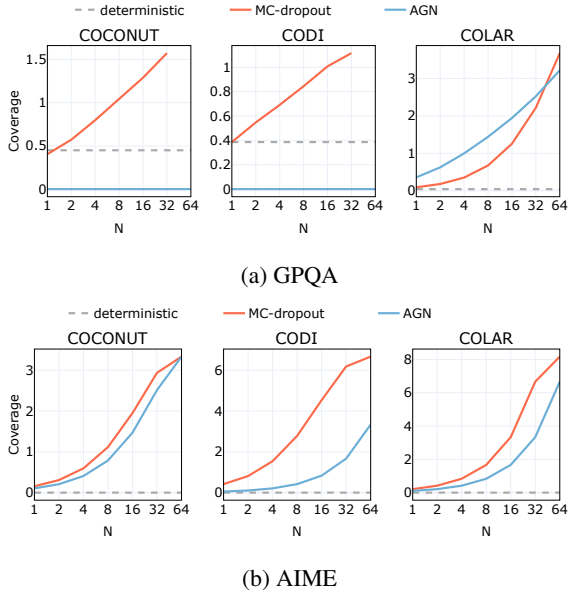


Figure 7: Scaling analysis on harder benchmarks (GPQA and AIME).

Method	GSM-Test	GSM-Hard	Compute
<b>NO-COT</b>			
deterministic	16.5	4.3	$\times 1$
Coverage@4	22.3	5.5	$\times 4$
<b>COT</b>			
deterministic	42.9	9.8	$\times 30^\dagger$
Coverage@4	50.7	11.6	$\times 120^\dagger$
<b>COCONUT</b>			
deterministic	34.0	7.0	$\times 6$
Coverage@4	<b>46.7</b>	<b>9.5</b>	$\times 24$
Best-of-4	34.5	7.5	$\times 24$

<sup>†</sup> Average compute across Test and Hard subsets.

Table 3: Performance and computational efficiency on GPT-2. Latent reasoning (COCONUT) achieves a superior balance between accuracy and computation compared to explicit CoT.

computational overhead by the number of decoding steps relative to the NO-COT baseline.

As summarized in Table 3, latent reasoning combined with TTS demonstrates superior efficiency, achieving competitive performance with significantly lower computational costs than explicit CoT. Specifically, COCONUT with TTS outperforms the explicit CoT baseline on the same backbone while requiring fewer decoding steps. Furthermore, while a performance gap persists between the Best-of- $N$  with LatentRM and the Coverage@ $N$  upper bound, integrating LatentRM provides consistent gains. These results suggest that latent reasoning, augmented by TTS, preserves the efficiency advantages of continuous-space processing while match-

Setting	Sampling	Agg.	Total
<b><math>N=1</math></b>			
COCONUT	0.050	—	<b>0.050</b>
COT	—	—	0.400
<b><math>N=32</math></b>			
COCONUT	1.322	0.386	<b>1.708</b>
COT	—	—	9.802

Table 4: Wall-clock latency comparison. Aggregation remains efficient even as  $N$  increases.

Method	Accuracy (%)
Deterministic	32.44
Best-of-4	33.81
Best-of-8	33.95
Best-of-16	34.04

Table 5: Performance with variable-length latent reasoning paths on GSM8K-Test.

ing or exceeding the accuracy of discrete reasoning.

## F.2 Wall-Clock Comparison of Latent and Explicit Reasoning with TTS

To complement the decoding-step comparison in Table 3, we further contrast COCONUT (latent reasoning) and explicit CoT under parallel Test-Time Scaling (TTS), using the settings summarized below. We measure end-to-end wall-clock latency in seconds per question on a single NVIDIA H100 GPU, and report sampling time, Best-of- $N$  aggregation time, and total latency.

As summarized in Table 4, COCONUT with TTS retains a favorable wall-clock profile relative to explicit CoT at matched  $N$ : aggregation contributes only a modest fraction of total time (on the order of 20% at  $N=32$ ), so parallel sampling rather than reranking dominates cost. The gap widens as  $N$  increases, consistent with short latent trajectories ( $T=6$  thoughts) versus long explicit chain-of-thought rollouts, together with sequence-level aggregation.

## F.3 Analysis on LatentRM under Variable-Length Setting

Latent reasoning models may generate trajectories of varying lengths, we thus conduct an evaluation of LatentRM in a variable-length setting. We modify the COCONUT paradigm by introducing stochasticity into the trajectory generation process, sampling the number of reasoning steps uniformly from 4 to 6, instead of fixing the trajectory length to 6. To mitigate potential length bias in trajectory selection, we utilize the mean step-wise reward, de-

fined as the total reward divided by the step count, to normalize scores. Experimental results in Table 5 indicate that best-of- $N$  reranking based on average reward consistently outperforms the deterministic baseline, suggesting that LatentRM can effectively handle variable-length latent reasoning.

## G Training Configuration for LatentRM

We train a Latent Reward Model (LatentRM) for COCONUT using the step-wise contrastive learning objective introduced in Section 4.2. model initialized from COOCNUT checkpoint, a GPT-2 backbone with 124 million parameters (Radford et al.), with architecture modifications detailed therein.

**Training data.** The training data consists of samples generated based on **GSM8K** training set (385K examples). For each input problem, we sample  $N = 8$  reasoning trajectories via MC-dropout with dropout probability  $p = 0.2$  from COCONUT, where  $p$  was tuned for optimal performance as mentioned in Section 6.1.

**Labeling.** For each intermediate reasoning step within a trajectory, we perform  $M = 128$  stochastic rollouts to empirically estimate the correctness of that step. trajectories that are either too easy (i.e., all trajectories are correct) or too difficult (i.e., all trajectories are incorrect) are excluded to improve training stability and focus on informative examples.

### Training configuration.

- Batch size: 2048
- Optimizer: Paged AdamW
- Learning rate:  $1 \times 10^{-5}$
- Epochs: 10
- Hardware:  $2 \times$  NVIDIA H100 GPUs

## H Algorithmic Specification of Latent Sampling Procedures

This section presents the pseudocode for the stochastic sampling procedures used in latent test-time scaling. We detail the inference-time execution of MC-Dropout and AGN for generating multiple latent reasoning trajectories.

---

### Algorithm 1 Sampling with MC-Dropout

---

**Require:** Input sequence  $x$ ; model  $f_{\theta}^{\text{latent}}$ ; dropout rate  $p$ ; sample count  $N$ ; stop-thinking condition  $\text{STOP}(\cdot)$ .

- 1: **Prefill:** Encode  $x$  and initialize kv-cache.
- 2: **for**  $n = 1$  to  $N$  **do**
- 3:    $t \leftarrow 0$
- 4:   **while** not  $\text{STOP}(h_{1:t}^{(n)})$  **do**
- 5:      $m_t^{(n)} \sim \text{Bernoulli}(1 - p)$     $\triangleright$  Sample dropout mask
- 6:      $\theta_t^{(n)} \leftarrow \theta \odot m_t^{(n)}$     $\triangleright$  Masked Subnetwork
- 7:      $h_{t+1}^{(n)} \leftarrow f_{\theta^{(n)}}(x, h_{1:t}^{(n)})$     $\triangleright$  Forward pass
- 8:      $t \leftarrow t + 1$
- 9:   **end while**
- 10:   Store trajectory  $\mathbf{h}^{(n)} \leftarrow [h_1^{(n)}, \dots, h_t^{(n)}]$
- 11:   Generate final answers  $y^{(n)} \leftarrow f_{\theta}(x, \mathbf{h}^{(n)})$
- 12: **end for**

---



---

### Algorithm 2 Sampling with AGN

---

**Require:** Input sequence  $x$ ; model  $f_{\theta}^{\text{latent}}$ ; noise scale  $\sigma$ ; sample count  $N$ ; stop-thinking condition  $\text{STOP}(\cdot)$ .

- 1: **Prefill:** Encode  $x$  and initialize kv-cache.
- 2: **for**  $n = 1$  to  $N$  **do**
- 3:    $t \leftarrow 0$
- 4:   Initialize latent state  $h_{1:0}^{(n)} \leftarrow \emptyset$
- 5:   **while** not  $\text{STOP}(h_{1:t}^{(n)})$  **do**
- 6:      $\epsilon_t^{(n)} \sim \mathcal{N}(0, \sigma^2 I)$     $\triangleright$  Sample Gaussian noise
- 7:      $\tilde{h}_t^{(n)} \leftarrow h_t^{(n)} + \epsilon_t^{(n)}$     $\triangleright$  Perturb latent thought
- 8:      $h_{t+1}^{(n)} \leftarrow f_{\theta}^{\text{latent}}(x, \tilde{h}_{1:t}^{(n)})$     $\triangleright$  Forward pass
- 9:      $t \leftarrow t + 1$
- 10:   **end while**
- 11:   Store trajectory  $\mathbf{h}^{(n)} \leftarrow [h_1^{(n)}, \dots, h_t^{(n)}]$
- 12:   Generate final answers  $y^{(n)} \leftarrow f_{\theta}(x, \mathbf{h}^{(n)})$
- 13: **end for**

---