

GTS: Inference-Time Scaling of Latent Reasoning with a Learnable Gaussian Thought Sampler

Minghan Wang¹, Ye Bai², Thuy-Trang Vu¹,
Ehsan Shareghi³, Gholamreza Haffari¹

¹Department of Data Science & AI, Monash University

²Faculty of Medicine Dentistry and Health Sciences, University of Melbourne

³Department of Computer Science, University College London

{minghan.wang, trang.vu1, gholamreza.haffari}@monash.edu

ye.bai2@student.unimelb.edu.au

ehsan.shareghi@ucl.ac.uk

Abstract

Inference-time scaling (ITS) for latent reasoning requires sampling multiple candidate latent trajectories, yet current methods typically rely on heuristic perturbations such as dropout or fixed Gaussian noise. We show that stronger perturbations do not necessarily improve candidate-set quality: they often induce larger distribution shifts without producing more decision-useful reasoning paths. This limitation arises because heuristic perturbations inject stochasticity through fixed or hand-designed mechanisms rather than an optimizable, context-conditioned sampling policy, making latent exploration difficult to control. We focus on the sampling side of latent ITS and propose the Gaussian Thought Sampler (GTS), a lightweight module that predicts context-dependent Gaussian perturbations over continuous reasoning states. Trained with GRPO-style policy optimization while keeping the backbone frozen, GTS turns heuristic perturbation into a trainable, context-conditioned sampling policy. Experiments across two latent reasoning backbones and multiple benchmarks show that GTS consistently improves pass@N over heuristic baselines, suggesting that effective sampling for latent ITS requires controlled, optimizable exploration rather than simply amplified stochasticity.¹

1 Introduction

Inference-time scaling (ITS) has emerged as a central mechanism for enhancing the reasoning performance of large language models (LLMs). By allocating additional test-time compute to two coupled stages: sampling multiple useful candidate trajectories and selecting or aggregating their decoded answers, approaches such as self-consistency and best-of- N sampling significantly improve accuracy without modifying model parameters (Wang et al.,

2023, 2024; Cobbe et al., 2021; Lightman et al., 2024). In discrete, token-based LLMs, sampling reasoning trajectories is naturally supported by explicit conditional distributions over next tokens. Strategies such as temperature scaling or nucleus sampling operate directly on these distributions, implicitly trading off diversity and likelihood under a probabilistic framework.

Recent advances in continuous latent reasoning introduce a different computational regime, where models refine hidden-state thought representations without generating intermediate textual tokens (Hao et al., 2024; Shen et al., 2025; Sui et al., 2025; Zhu et al., 2025). While efficient and expressive, this paradigm removes the token-level distributions that support principled sampling in discrete models. As a result, extending the conventional sample-then-select ITS mechanism to intermediate latent thoughts is non-trivial: on the sampling side, existing latent reasoning models typically rely on heuristic perturbations, such as dropout or fixed Gaussian noise (Wang et al., 2025; You et al., 2026). These perturbations introduce stochasticity, but do not define an explicit sampling distribution over latent thoughts.

We therefore focus on this under-defined sampling component and revisit latent ITS through the lens of *exploration over reasoning trajectories* at inference time², which raises a fundamental question: **what constitutes effective exploration in continuous reasoning space?** Our analysis (§2 and §5) shows that the challenge is not simply how much perturbation to inject, but how to control exploration. Heuristic perturbations use manually chosen global scales that are sensitive to the backbone, task, and latent geometry, making them prone to under- or over-exploration (i.e., overly concentrated or excessively diffuse search); they may increase diversity or distribution shift without mov-

¹Our code is available at: <https://github.com/yuriak/GaussianThoughtSampler>

²We formalize this notion of exploration in §2.1.

ing samples toward decision-improving regions, as illustrated in Figure 1.

To address this limitation, we introduce the **Gaussian Thought Sampler (GTS)**, a lightweight module that predicts context-dependent Gaussian perturbations over latent reasoning states while keeping the backbone frozen. By defining an explicit conditional density, GTS turns heuristic noise injection into a trainable sampling policy for latent exploration, bringing latent ITS closer to the distributional perspective that underlies token-level inference-time scaling. Our contributions are as follows:

- We identify a key limitation of heuristic latent ITS: effective scaling requires controlling sampling quality, not just injecting stochasticity.
- We propose GTS, a lightweight conditional Gaussian sampler that makes latent perturbation explicit and optimizable.
- We demonstrate consistent gains over heuristic perturbation baselines on two representative latent reasoning backbones across in- and out-of-distribution arithmetic benchmarks, with additional evidence on commonsense and multi-step logical reasoning tasks.

2 Diagnostic Analysis of Heuristic Sampling

2.1 Exploration as Trajectory Sampling in ITS

ITS improves reasoning by allocating additional test-time compute to sample multiple candidate trajectories and select or aggregate their decoded answers (Wang et al., 2023). This resembles *exploration* in reinforcement learning: a stochastic policy samples trajectories, and performance improves when the sampled set contains better outcomes (Sutton and Barto, 1998; Stiennon et al., 2022; DeepSeek-AI et al., 2025). We therefore formalize ITS as **exploration over reasoning trajectories** at inference time. Given an input x , let $\mathcal{T}(x)$ denote the set of valid reasoning trajectories and let $\pi(\tau \mid x)$ be the inference-time sampling policy. Under an **exploration budget** of N , ITS draws

$$\tau^{(1)}, \dots, \tau^{(N)} \sim \pi(\cdot \mid x), \quad \tau^{(i)} \in \mathcal{T}(x), \quad (1)$$

and predicts by selecting or aggregating answers decoded from these trajectories. Unlike RL, this

exploration is used only to generate candidates and does not update π during inference.

This view highlights a key difference between text and latent reasoning. In text-based models, $\pi(\tau \mid x)$ is induced by token-level stochastic decoding from explicit categorical distributions. In latent reasoning models, intermediate reasoning steps are typically deterministic,

$$h_{t+1}^{\text{det}} = f_{\theta}(h_t^{\text{det}}, x), \quad (2)$$

and therefore do not directly define a stochastic trajectory distribution (where f_{θ} is the LLM and h^{det} is the deterministic hidden state). Existing methods enable exploration by perturbing latent dynamics, for example

$$\tilde{h}_{t+1} = f_{\theta}(\tilde{h}_t, x) + z_t, \quad (3)$$

where z_t is a stochastic perturbation.

Under finite budgets, the central issue is not merely whether sampling produces different trajectories, but whether it produces *useful* ones. We therefore evaluate **exploration quality**: whether heuristic perturbations move the model’s predictive belief toward the correct answer, rather than merely introducing random variation.

2.2 Experimental Setup

Models We evaluate a text-based reasoning model, i.e., GPT-2 (Radford et al., 2019) fine-tuned on GSM8K-Aug (Deng et al., 2023), and a latent reasoning model, COCONUT, built on the same backbone. GPT-2 produces textual reasoning followed by the delimiter "###" before answer generation. COCONUT performs $K = 6$ latent reasoning steps, following prior implementation details (Hao et al., 2024).

Sampling Protocol For each input x in the GSM8K-test (Cobbe et al., 2021) dataset, we generate $N = 32$ reasoning trajectories by applying sampling only to the reasoning stage. For each trajectory τ , we quantify the predictive probability of the first ground-truth answer token y_1^* using teacher forcing after appending the answer prefix. Since GSM8K answers are numeric, we focus on y_1^* to reduce multi-token noise while preserving decision information. GPT-2 uses token-level sampling (temperature 1.0) and dropout sampling ($p \in \{0.1, 0.5\}$). COCONUT applies dropout-based perturbations in latent space.

Sampling	SG $\uparrow$	SG $> 0.5 \uparrow$	JS
GPT-2			
Token (temp=1.0)	9.94	0.62	0.29
Dropout ($p = 0.1$)	9.64	0.58	0.30
Dropout ($p = 0.5$)	3.89	0.57	0.67
COCONUT			
Dropout ($p = 0.1$)	1.09	0.61	0.05
Dropout ($p = 0.5$)	-0.87	0.40	0.28

Table 1: Preliminary sampling quality analysis on GPT-2 and COCONUT. We report SG, SG rate, and distribution shift (JS). Higher is better for SG and SG rate.

2.3 Measuring Sampling Quality

Let τ^{det} denote the deterministic reasoning trajectory, and let $p(y_1^* | x, \tau)$ be the predictive probability of the first correct answer token.

Sampling Gain (SG) We define trajectory-level gain as the change in log-odds of the correct answer:

$$\Delta(\tau) = s(\tau) - s(\tau^{\text{det}}), \quad (4)$$

$$s(\tau) = \log \frac{p(y_1^* | x, \tau)}{1 - p(y_1^* | x, \tau)} \quad (5)$$

Positive $\Delta(\tau)$ indicates improved decision confidence relative to the deterministic baseline. To reflect best-of- N selection, we define

$$\text{SG}(x) = \max_{k \leq N} \Delta(\tau_k), \quad (6)$$

and report the dataset-level mean SG.

Sampling Gain Rate We also report the fraction of inputs with $\text{SG}(x) > 0.5$, corresponding to a substantial increase in decision odds.

Distribution Shift We measure distribution shift by the Jensen–Shannon divergence (JS) between answer-token distributions:

$$\text{JS}(\tau) = \text{JS}\left(p(\cdot | x, \tau) \parallel p(\cdot | x, \tau^{\text{det}})\right). \quad (7)$$

Mean JS captures how strongly sampling alters the answer distribution, regardless of whether the change is beneficial.

2.4 Results

Table 1 reveals a consistent mismatch between distribution shift and sampling quality. For GPT-2, token sampling and mild dropout both achieve high SG with moderate JS, whereas stronger dropout

greatly increases JS but substantially reduces SG. The effect is more pronounced in COCONUT: mild latent dropout yields positive SG with minimal shift, while stronger dropout produces negative SG despite larger divergence.

These results show that stronger perturbations can move samples farther from the deterministic trajectory without making them more useful. This mismatch is especially pronounced in latent reasoning, where heuristic perturbations are more likely to move sampling away from decision-improving regions. These results suggest that effective latent exploration requires more than simply increasing stochasticity, and instead calls for better-controlled sampling mechanisms.

3 Gaussian Thought Sampler

3.1 Overview

We augment a frozen latent reasoning backbone with a learnable, context-conditioned sampling policy over hidden-state perturbations. Instead of injecting heuristic noise into deterministic latent reasoning, GTS models exploration as an explicit conditional density over continuous perturbations. This reformulates inference-time exploration as a probabilistic modeling problem and enables direct optimization of the sampling mechanism. Unlike latent-variable or per-instance optimization methods (Kong et al., 2025; Zhou et al., 2025; Chen et al., 2024), GTS learns an amortized sampling policy for existing deterministic latent reasoning backbones, see Appendix A for further discussion.

3.2 Conditional Latent Sampling

Problem Setup Given an input question x , a latent reasoning model parameterized by θ performs K latent reasoning steps and then produces an answer distribution:

$$p_\theta(y | x, \mathbf{h}_{1:K}) \quad (8)$$

where $\mathbf{h}_k \in \mathbb{R}^d$ denotes the backbone hidden state at step k . Our goal is to introduce a learnable conditional sampling distribution over latent perturbations while keeping θ fixed.

Perturbation Variable We introduce a continuous perturbation variable $\mathbf{z}_k \in \mathbb{R}^d$ at each reasoning step and define:

$$\tilde{\mathbf{h}}_k = \mathbf{h}_k^{\text{det}} + \mathbf{z}_k, \quad (9)$$

where $\mathbf{h}_k^{\text{det}}$ is the deterministic hidden state. The perturbed state $\tilde{\mathbf{h}}_k$ is fed back into the backbone

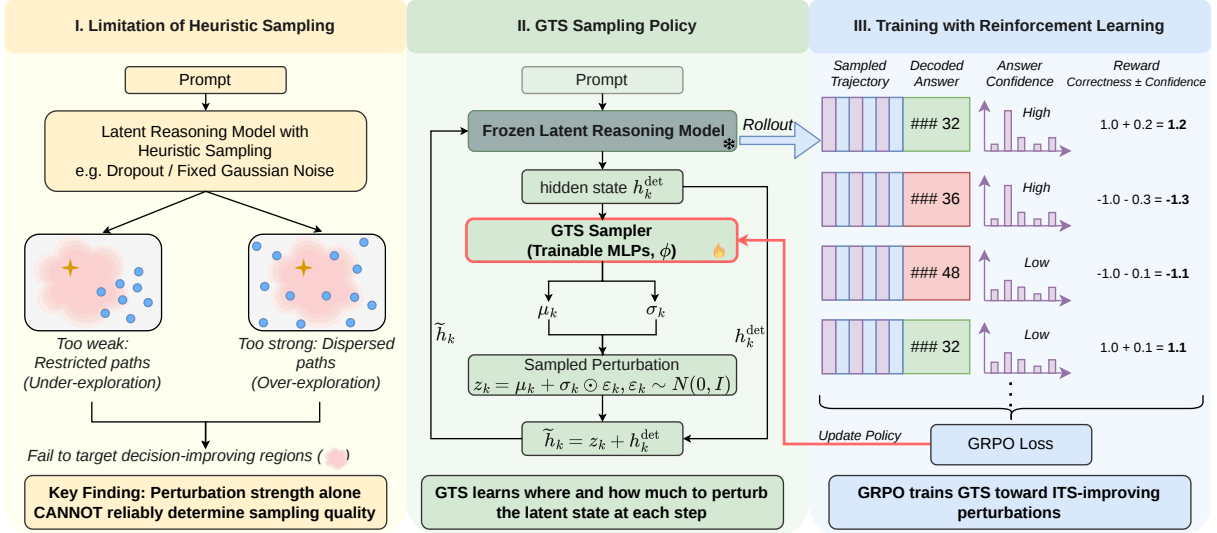


Figure 1: Overview of GTS. **Left:** Heuristic latent perturbations cannot reliably control sampling quality: weak noise under-explores, while strong noise over-disperses trajectories. **Middle:** GTS predicts a context-dependent Gaussian perturbation distribution from the deterministic latent state at each step. **Right:** Trajectories are scored by correctness and confidence, and GRPO trains GTS to favor perturbations that improve inference-time scaling.

for subsequent reasoning, yielding a sampled latent trajectory: $\tau = \{\tilde{\mathbf{h}}_1, \dots, \tilde{\mathbf{h}}_K\}$.

Context-Conditioned Gaussian Policy We parameterize a conditional Gaussian policy over perturbations:

$$q_\phi(\mathbf{z}_k | \mathbf{c}_k) = \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{c}_k), \text{diag}(\boldsymbol{\sigma}_\phi^2(\mathbf{c}_k))), \quad (10)$$

where $\mathbf{c}_k$ denotes the conditioning context at step k (in practice, the backbone hidden state $\mathbf{h}_k^{\text{det}}$). Sampling follows the reparameterization:

$$\mathbf{z}_k = \boldsymbol{\mu}_\phi(\mathbf{c}_k) + \boldsymbol{\sigma}_\phi(\mathbf{c}_k) \odot \boldsymbol{\epsilon}_k, \quad \boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \quad (11)$$

Because Equation (9) defines an affine transformation with unit Jacobian, the change of variables preserves density. Therefore, learning $q_\phi(\mathbf{z}_k | \mathbf{c}_k)$ is equivalent to learning an explicit conditional density over perturbed thought representations $\tilde{\mathbf{h}}_k$. The backbone computation remains unchanged, while GTS governs how perturbations are sampled during latent reasoning.

3.3 Policy Learning

GTS defines an explicit conditional density over latent perturbation trajectories. A natural alternative would be likelihood-based training, e.g., ELBO-style variational objectives (Kingma and Welling, 2022). However, our goal is not to model a latent posterior, but to directly optimize inference-time exploration under task-level rewards. The quality

of a perturbation trajectory is determined by task-level outcomes, which are non-differentiable with respect to the sampler parameters. We therefore treat latent perturbations as continuous actions and optimize the policy via reinforcement learning.

Trajectory Policy For an input x , a perturbation trajectory $\tau = \{\mathbf{z}_1, \dots, \mathbf{z}_K\}$ defines a factorized Gaussian policy:

$$\log q_\phi(\tau | x) = \sum_{k=1}^K \log q_\phi(\mathbf{z}_k | \mathbf{c}_k). \quad (12)$$

For a diagonal Gaussian, the per-step log-density admits a closed-form expression:

$$\begin{aligned} \log q_\phi(\mathbf{z}_k | \mathbf{c}_k) = & \\ & - \frac{1}{2} \sum_{d=1}^D \left[\left(\frac{z_{k,d} - \mu_{k,d}}{\sigma_{k,d}} \right)^2 + 2 \log \sigma_{k,d} + \log(2\pi) \right]. \end{aligned} \quad (13)$$

This closed-form density is essential: it enables exact computation of policy likelihoods and density ratios, making policy-gradient optimization well-defined in continuous latent space.

Reward Design For each input x , we sample N trajectories and decode answers $\{a^{(i)}\}_{i=1}^N$ during rollout. Let y^* denote the ground-truth answer. The reward for trajectory i is defined as

$$r^{(i)} = r_0(2\mathbb{I}[a^{(i)} = y^*] - 1) + \alpha s^{(i)}, \quad (14)$$

where the first term provides a symmetric correctness signal and $s^{(i)}$ is a confidence-based shaping term derived from the normalized log-probability of the generated answer. The shaping term encourages high-confidence correct trajectories and discourages high-confidence incorrect ones, while remaining secondary to correctness. See Appendix B.1 for more details.

GRPO-Style Policy Optimization To stabilize policy updates, we maintain a reference sampler $q_{\phi_{\text{ref}}}$ as an exponential moving average of the current policy (Schulman et al., 2017; Ouyang et al., 2022). For trajectory $\tau^{(i)}$, we compute the density ratio:

$$\rho^{(i)} = \frac{q_{\phi}(\tau^{(i)} | x)}{q_{\phi_{\text{ref}}}(\tau^{(i)} | x)}. \quad (15)$$

In implementation, we compute this ratio using a dimension-normalized log-ratio surrogate for numerical stability in high-dimensional latent space; see Appendix B.2. The analytical Gaussian density still provides the underlying closed-form quantity for both the current and reference policies. Following GRPO-style clipped optimization (DeepSeek-AI et al., 2025), the policy gradient objective is:

$$\mathcal{L}_{\text{PG}} = -\mathbb{E} \left[\min(\rho^{(i)} A^{(i)}, \text{clip}(\rho^{(i)}, 1 - \epsilon_c, 1 + \epsilon_c) A^{(i)}) \right], \quad (16)$$

where $A^{(i)}$ denotes the group-normalized advantage computed within each prompt, and $1 \pm \epsilon_c$ defines the clipping range. We further regularize the sampler via a KL penalty between the current and reference Gaussian policies:

$$\mathcal{L}_{\text{KL}} = \beta \mathbb{E} [\text{KL}(q_{\phi}(\mathbf{z}_k | \mathbf{c}_k) \| q_{\phi_{\text{ref}}}(\mathbf{z}_k | \mathbf{c}_k))], \quad (17)$$

which also admits a closed-form solution. The final objective is:

$$\mathcal{L}_{\text{GTS}} = \mathcal{L}_{\text{PG}} + \mathcal{L}_{\text{KL}}. \quad (18)$$

This formulation directly optimizes the inference-time exploration distribution in continuous latent space while leaving the base language model unchanged.

Design Rationale This diagonal Gaussian formulation provides a minimal tractable form of continuous latent sampling. It defines an explicit conditional density, enabling exact log-probabilities, density ratios, and KL regularization for policy optimization. Its context-dependent mean and variance

support both directed latent steering and adaptive stochastic exploration, while keeping the sampler lightweight and the backbone unchanged. See Appendix B.3 for additional discussion.

4 Experiments

4.1 Experimental Setup

Data We use the **GSM8K-aug** training corpus adopted in prior latent reasoning work (Deng et al., 2023). We use answer-focused mathematical reasoning as our primary controlled setting because its verifiable rewards, exact-match evaluation, and standard pass@ N diagnostics provide a clean setting for isolating sampling quality. From the full augmented set (386k), we uniformly sample 20k instances for training. Evaluation is performed on the standard GSM8K (Cobbe et al., 2021) test set (1319 samples). To assess transfer beyond the training distribution within arithmetic reasoning, we additionally evaluate the same trained samplers on three out-of-distribution benchmarks: **Multi-Arith** (Roy and Roth, 2015), **SVAMP** (Patel et al., 2021), and **GSM8K-Hard** (Gao et al., 2023). No additional training or hyperparameter tuning is performed for these datasets; all samplers are applied directly at inference time under the same settings as the main experiments. To examine whether the observed sampling behavior extends beyond arithmetic reasoning, we further evaluate on three non-arithmetic benchmarks: **CommonsenseQA** (Talmor et al., 2019; Shen et al., 2025) for commonsense reasoning, and **ProsQA** (Hao et al., 2024) and **ProntoQA** (Saparov and He, 2023) for multi-step logical reasoning. These experiments use task-specific latent reasoning backbones and are therefore reported separately from the arithmetic out-of-distribution evaluation.

Models We evaluate GTS on two representative latent reasoning models with different architectures and latent dynamics: COCONUT (Hao et al., 2024) and CODI (Shen et al., 2025). For COCONUT, we follow Hao et al. (2024), using a GPT-2-based recurrent latent reasoning backbone with $K = 6$ latent steps. For CODI, we use a larger LLAMA-3.2-1B (Grattafiori et al., 2024) backbone with 6 latent reasoning steps and its recurrent filtering module. Together, they test whether a learnable perturbation policy improves sampling over heuristic perturbations across distinct latent reasoning regimes. In all cases, backbone parameters are frozen and only the Gaussian sampler

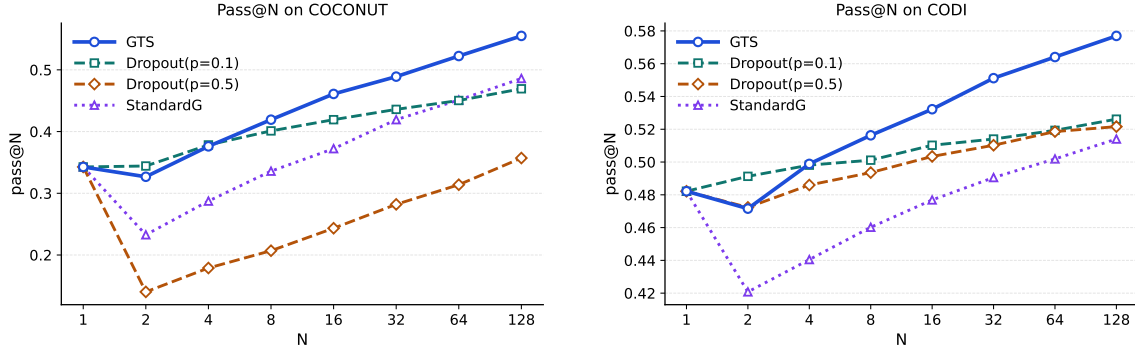


Figure 2: ITS performance under different sampling strategies. Pass@ N on COCONUT (left) and CODI (right) as a function of the sampling budget N . All methods coincide at $N = 1$, corresponding to deterministic latent reasoning. As N increases, GTS scales more consistently than dropout-based sampling and standard Gaussian noise (StandardG). Additional out-of-distribution results are reported in §C.1.

is trained. For the non-arithmetic evaluation, we use publicly released COCONUT checkpoints for ProsQA and ProntoQA (Dilgren and Wiegrefe, 2026). For CommonsenseQA, we train a GPT-2-based CODI backbone using the official CODI codebase³. GTS is then trained on each frozen task-specific backbone under the same protocol as the main experiments.

GTS Architecture The sampler consists of lightweight mean and log-standard-deviation heads parameterizing a diagonal Gaussian policy over latent perturbations. To avoid premature collapse to a deterministic policy, we enforce a minimum log-standard-deviation during training ($\log \sigma > -2.0$). Latent perturbations are injected only during recursive latent reasoning steps (from `<start_latent>` to `<lat_5>` when $K = 6$), ensuring that stochasticity only affects the reasoning dynamics rather than the final answer decoding stage. Additional implementation details are provided in Appendix B.2.

Training For each prompt, we sample $N = 32$ perturbation trajectories during rollout. Unless otherwise stated, training runs for 10K optimization steps with batch size set as 32 and a learning rate of 1×10^{-4} with linear warmup. Other hyperparameters can be found in Appendix B.2.

Baselines We compare GTS with two stochastic inference baselines: **Dropout Sampling**, which enables dropout during latent reasoning with $p \in \{0.1, 0.5\}$ while disabling it for prompt prefilling and answer generation; and **Standard Gaussian Noise**, which adds isotropic noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ to each latent hidden state without context condition-

ing or learned parameters.

Evaluation We evaluate ITS performance by varying the sampling budget $N \in \{1, 2, 4, 8, 16, 32, 64, 128\}$. When $N = 1$, all methods reduce to deterministic inference. For $N \geq 2$, stochasticity is applied exclusively to latent reasoning, and answers are decoded **greedily** to isolate the effect of thought sampling. We use pass@ N as a candidate-set quality metric: it measures whether the sampler can expose at least one correct trajectory, rather than evaluating a particular non-oracle aggregation rule. For the non-arithmetic experiments, we report pass@ N at $N \in \{8, 32, 128\}$; CommonsenseQA results are reported as mean $\pm$ standard deviation over three random seeds.

4.2 Main Results

Overall scaling behavior Figure 2 shows pass@ N curves on COCONUT and CODI. At $N = 1$, all methods reduce to deterministic latent reasoning. As N increases, stochastic latent sampling generally improves candidate-set quality, confirming that latent thoughts can support inference-time scaling. The curves also reveal a small-budget exploration cost: GTS and the standard Gaussian noise baseline (StandardG) can slightly reduce performance at $N = 2$, suggesting that perturbations may initially disrupt individual trajectories before the benefit of a larger candidate set becomes apparent.

Comparison with baselines Heuristic perturbation methods show substantially different behaviors across settings. Dropout with a mild rate ($p = 0.1$) provides stable gains on both models

³<https://github.com/zhenyi4/codi>

Method	@8	@32	@128
CommonsenseQA (Det. 39.3)			
Dropout	43.4±0.16	46.7±0.29	50.5±0.17
StandardG	43.9±0.13	45.9±0.05	48.9±0.45
GTS	46.7±0.05	51.3±0.17	57.5±0.19
ProsQA (Det. 98.0)			
Dropout	98.0	98.2	98.0
StandardG	98.0	98.0	98.0
GTS	98.0	98.4	98.4
ProntoQA (Det. 99.0)			
Dropout	99.1	99.1	99.1
StandardG	99.1	99.1	99.3
GTS	99.1	99.3	99.3

Table 2: Pass@ N (%) on non-arithmetic reasoning benchmarks. Deterministic accuracy is shown in parentheses. CommonsenseQA results are mean $\pm$ standard deviation over three random seeds.

and remains a relatively strong baseline, especially at small-to-moderate budgets. In contrast, stronger dropout ($p = 0.5$) consistently performs worse, most clearly on COCONUT, where it causes a sharp small-budget degradation and remains well below the other methods throughout. StandardG is also less stable, with weaker small-budget behavior and less consistent scaling than mild dropout. Taken together, these results suggest that fixed heuristic perturbations can help, but their effectiveness is highly sensitive to perturbation strength and model dynamics.

Effectiveness GTS achieves the strongest scaling behavior at moderate-to-large budgets on both backbones. Its advantage over heuristic baselines becomes clearer as N grows, indicating that the learned sampler produces more useful candidate trajectories under finite inference budgets. The larger relative gain on COCONUT suggests that the benefit of learned perturbation control depends on backbone-specific latent dynamics; we discuss this further in Appendix D.1. Appendix C.1 shows the same overall pattern on three out-of-distribution benchmarks, with GTS remaining more consistent than heuristic perturbations. Overall, these arithmetic experiments show that learning a context-conditioned perturbation policy yields more effective sampling-side latent ITS than globally fixed stochastic perturbations.

Generalization beyond arithmetic Table 2 extends the evaluation to commonsense and multi-step logical reasoning. On CommonsenseQA, GTS

shows clear scaling gains as the sampling budget increases, improving pass@ N from 46.7% at $N = 8$ to 57.5% at $N = 128$. At $N = 128$, this substantially exceeds both Dropout (50.5%) and StandardG (48.9%), showing that the benefit of learned latent sampling is not restricted to arithmetic reasoning. ProsQA and ProntoQA provide a complementary sanity check: their deterministic latent backbones already achieve 98.0% and 99.0%, respectively, leaving little headroom for inference-time scaling. GTS preserves this near-saturated performance and reaches 98.4% on ProsQA and 99.3% on ProntoQA at larger sampling budgets. We therefore interpret these two benchmarks primarily as evidence that the learned sampler does not degrade already near-ceiling logical reasoning performance, rather than as settings with substantial scaling headroom.

Controlled follow-up studies Appendix C further examines robustness and scope. Perturbation sweeps (App. C.2) show that heuristic baselines remain sensitive to tuning, with optimal scales varying across backbones and budgets. A norm-adaptive Gaussian control further shows that simple state-dependent rescaling does not close the gap to GTS. Latent-step experiments (App. C.3) show that GTS remains effective under moderate trajectory-length changes. Repeated evaluation over five random seeds (App. C.7) confirms that GTS maintains consistent gains with low variance across stochastic rollouts. Token sampling experiments (App. C.4) further show that the improvements are not tied to greedy answer decoding. Under simple non-oracle selection rules (App. C.5), GTS also improves selected-answer accuracy, although the gap to pass@ N confirms that answer selection remains a separate bottleneck. Reward ablations (App. C.8) further show that dense shaping improves candidate-set quality over an accuracy-only variant. Computational profiling (App. C.9) further shows that GTS adds only modest latency, memory, and parameter overhead.

5 Analysis

5.1 On Sampling Quality

We revisit sampling quality using the diagnostic metrics introduced in §2. Following the same protocol and dataset, we evaluate SG, SG rate, and distribution shift on both COCONUT and CODI, together with StandardG as additional baseline consistent with the ITS setup in §4.1.

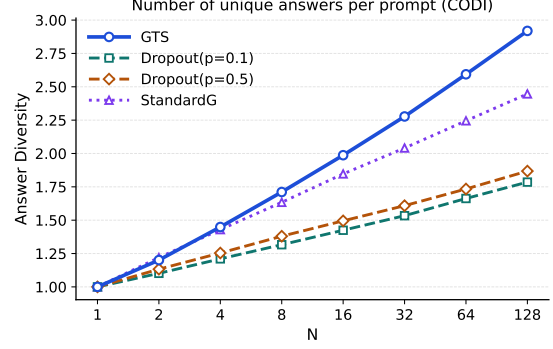
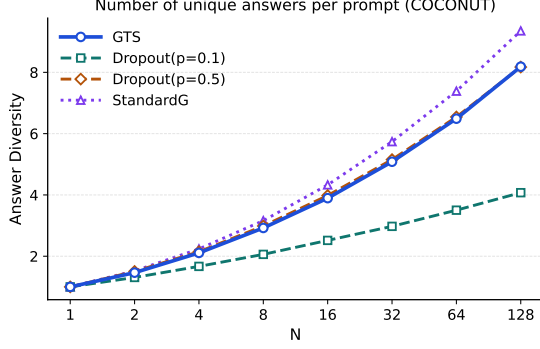


Figure 3: Average number of unique decoded answers per prompt.

Sampling	SG $\uparrow$	SG $> 0.5 \uparrow$	JS
COCONUT			
Dropout ($p = 0.1$)	1.09	0.61	0.05
Dropout ($p = 0.5$)	-0.87	0.40	0.28
StandardG	0.34	0.51	0.20
GTS	1.84	0.82	0.11
CODI			
Dropout ($p = 0.1$)	0.53	0.31	0.01
Dropout ($p = 0.5$)	0.47	0.39	0.06
StandardG	0.11	0.38	0.15
GTS	1.15	0.70	0.10

Table 3: Sampling quality analysis for latent reasoning models on GSM8K. We report SG, SG rate, and distribution shift (measured by JS divergence). Higher is better for SG and SG rate.

Failure modes of heuristic perturbations Table 3 shows that the mismatch between distribution shift and sampling quality observed in §2 persists across both latent reasoning backbones. We summarize this behavior with two recurring regimes:

- **Under-exploration** refers to perturbations that are too weak to move sampling meaningfully away from the deterministic trajectory, yielding limited distribution shift and only modest gains.
- **Over-exploration** refers to perturbations that are too strong and disrupt decision-relevant information, producing large distribution shift but limited or even negative SG and reduced SG rate.

These regimes are clearly reflected in Table 3 and heuristic perturbation sensitivity analysis in Appendix C.2. For COCONUT, mild dropout remains in the under-exploration regime, while stronger dropout moves into over-exploration, sharply increasing distribution shift but driving SG negative. For CODI, the same pattern appears more mildly: stronger perturbations again increase

distribution shift without proportional gains in sampling quality. Across both backbones, GTS avoids these extremes and achieves positive SG with relatively controlled distribution shift. Appendix D.3 provides a candidate-set interpretation of this effect, showing why finite-budget scaling depends on assigning probability mass to useful latent trajectories rather than merely increasing perturbation magnitude.

5.2 On Sampling Behavior

Answer-level diversity Figure 3 reports the average number of unique decoded answers per prompt, capturing discrete branching after greedy decoding. Unlike distribution shift in Table 3, higher answer diversity does not necessarily imply stronger SG or better pass@ N scaling. StandardG often produces more unique answers than GTS at moderate budgets, yet this additional branching does not translate into higher SG or pass@ N , showing that output diversity alone is not necessarily decision-useful.

Step-wise signal-to-noise ratio To analyze how stochastic intervention evolves across latent reasoning steps, we measure a step-wise signal-to-noise ratio (SNR) on GSM8K-test, sampling $N = 32$ trajectories per prompt and averaging SNR at the prompt level. At latent step t , with the predicted mean vector $\mu_t \in \mathbb{R}^D$ and diagonal log standard deviation $\log \sigma_t \in \mathbb{R}^D$, we define:

$$\text{SNR}_t = \frac{\sqrt{\frac{1}{D} \|\mu_t\|_2^2}}{\sqrt{\frac{1}{D} \sum_{i=1}^D \sigma_{t,i}^2}}. \quad (19)$$

SNR measures the relative strength of deterministic steering and injected noise; values below 1 indicate noise-dominated steps. Figure 4 shows that COCONUT exhibits a saw-tooth, moderate-SNR pattern across adjacent latent steps, while

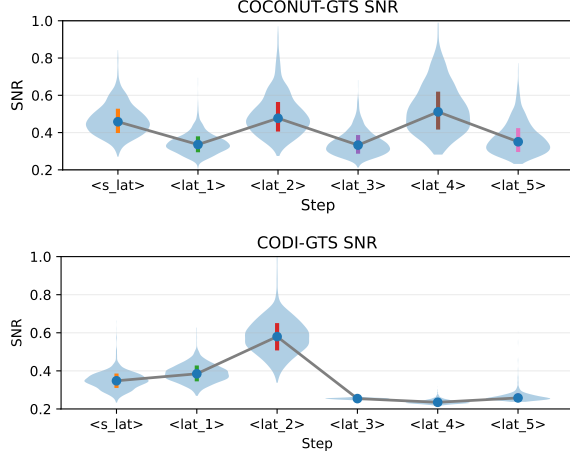


Figure 4: Step-wise distribution of signal-to-noise ratio (SNR) across latent reasoning steps. Each violin shows the distribution over prompts; markers denote medians.

N	h	$h + \mu$	$h + \sigma \odot \epsilon$	$h + z$
COCONUT				
8	34.2	35.9	37.1	42.2
32	34.2	35.9	44.4	49.6
128	34.2	35.9	51.3	55.6
CODI				
8	48.1	49.7	48.9	51.8
32	48.1	49.7	50.8	55.3
128	48.1	49.7	53.1	57.5

Table 4: Decomposing GTS perturbations on GSM8K. We report pass@ N (%) for deterministic states h , mean-only $h + \mu$, variance-only $h + \sigma \odot \epsilon$, and full GTS $h + z$, where $\epsilon \sim \mathcal{N}(0, I)$ and $z = \mu + \sigma \odot \epsilon$.

CODI concentrates deterministic steering early (peaking at <lat_2>) and shifts to lower-SNR exploration later. Thus, GTS does not impose a fixed stochastic schedule, but adapts the steering-noise balance to the backbone’s latent dynamics (See Appendix D.2 for further discussion). Appendix C.6 further shows that the learned perturbations are bounded and state-dependent rather than simple rescalings of latent states.

5.3 Disentangling Steering and Stochasticity

To understand how GTS uses its Gaussian parameterization, we decompose the learned perturbation into its mean and variance components in Table 4. Mean-only perturbations provide limited gains, indicating that deterministic steering alone cannot account for the scaling improvement. In contrast, variance-only perturbations already improve with larger sampling budgets, showing that learned stochasticity contributes useful candidate

diversity. The full perturbation consistently performs best, suggesting that effective latent exploration benefits from coupling directional control with adaptive uncertainty. This decomposition explains why a context-conditioned Gaussian is a suitable minimal policy class: the mean provides reusable directional corrections in latent space, while the variance controls where sampling should remain exploratory under finite inference budgets.

6 Conclusion

We study inference-time scaling in latent reasoning models from the perspective of sampling in continuous thought space. Our analysis shows that heuristic perturbations do not reliably produce effective exploration: larger distribution shift or higher answer diversity does not necessarily translate into better sampling quality, and fixed perturbation schemes can easily fall into under- or over-exploration. To address this limitation, we introduce GTS as a proof of concept, with a lightweight Gaussian sampler that models latent perturbation as a conditional sampling policy. Across two representative latent reasoning architectures, GTS yields stronger and more reliable scaling on arithmetic benchmarks and extends these gains to additional commonsense and logical reasoning settings, with ablations showing that both directed steering and learned stochasticity contribute to its gains. Overall, our results suggest that effective sampling-side latent ITS requires not just more stochasticity or diversity, but better-controlled and optimizable exploration over latent reasoning trajectories.

Limitations

This work has several limitations. GTS may be less effective under very small sampling budgets, noisy or unavailable reward signals, or substantially different latent geometries where a diagonal perturbation policy is insufficient. In addition, our main focus is on the sampling component of latent ITS: GTS aims to improve the generation of candidate latent reasoning trajectories, while final answer selection or aggregation remains a complementary direction. More advanced verifiers, process reward models, or latent reward models may further improve end-to-end inference-time scaling when combined with stronger latent candidate generation. Our analysis of sampling behavior remains empirical and does not provide a formal theoretical characterization of exploration in high-dimensional

latent spaces. Finally, we study two representative latent reasoning architectures, and the behavior of learnable perturbation policies may differ under alternative continuous reasoning formulations. We leave broader task coverage, sampler-selector co-design, and theoretical analysis to future work.

Acknowledgments

The authors are grateful to the anonymous reviewers for their helpful comments. This research/work was supported by Monash eResearch capabilities, including M3.

References

- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. [Graph of thoughts: Solving elaborate problems with large language models](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.
- Haolin Chen, Yihao Feng, Zuxin Liu, Weiran Yao, Akshara Prabhakar, Shelby Heinecke, Ricky Ho, Phil Mui, Silvio Savarese, Caiming Xiong, and Huan Wang. 2024. [Language models are hidden reasoners: Unlocking latent reasoning capabilities via self-rewarding](#). *Preprint*, arXiv:2411.04282.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, and Henrique Ponde de Oliveira Pinto et al. 2021. [Evaluating large language models trained on code](#). *Preprint*, arXiv:2107.03374.
- Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. 2025. [Pass@k training for adaptively balancing exploration and exploitation of large reasoning models](#). *Preprint*, arXiv:2508.10751.
- Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. [Reasoning with exploration: An entropy perspective](#). *Preprint*, arXiv:2506.14758.
- Jeffrey Cheng and Benjamin Van Durme. 2024. [Compressed chain of thought: Efficient reasoning through dense representations](#). *Preprint*, arXiv:2412.13171.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *Preprint*, arXiv:2110.14168.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, and Junxiao Song et al. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *Preprint*, arXiv:2501.12948.
- Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. 2023. [Implicit chain of thought reasoning via knowledge distillation](#). *Preprint*, arXiv:2311.01460.
- Connor Dilgren and Sarah Wiegrefe. 2026. [Are latent reasoning models easily interpretable?](#) *Preprint*, arXiv:2604.04902.
- Yarin Gal and Zoubin Ghahramani. 2016. [Dropout as a bayesian approximation: Representing model uncertainty in deep learning](#). *Preprint*, arXiv:1506.02142.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. [Pal: Program-aided language models](#). *Preprint*, arXiv:2211.10435.
- Halil Alperen Gozeten, M. Emrullah Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. 2025. [Continuous chain of thought enables parallel exploration and reasoning](#). *Preprint*, arXiv:2505.23648.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and Ahmad Al-Dahle et al. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- 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](#). *Preprint*, arXiv:2412.06769.
- Diederik P Kingma and Max Welling. 2022. [Auto-encoding variational bayes](#). *Preprint*, arXiv:1312.6114.
- Deqian Kong, Minglu Zhao, Dehong Xu, Bo Pang, Shu Wang, Edouardo Honig, Zhangzhang Si, Chuan Li, Jianwen Xie, Sirui Xie, and Ying Nian Wu. 2025. [Latent thought models with variational bayes inference-time computation](#). *Preprint*, arXiv:2502.01567.
- Hengli Li, Chenxi Li, Tong Wu, Xuekai Zhu, Yuxuan Wang, Zhaoxin Yu, Eric Hanchen Jiang, Song-Chun Zhu, Zixia Jia, Ying Nian Wu, and Zilong Zheng. 2026. [Seek in the dark: Reasoning via test-time instance-level policy gradient in latent space](#). *Preprint*, arXiv:2505.13308.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. [Let’s verify step by step](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#). *Preprint*, arXiv:2203.02155.

- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. [Are NLP models really able to solve simple math word problems?](#) In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*.
- Subhro Roy and Dan Roth. 2015. [Solving general arithmetic word problems](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.
- Abulhair Saparov and He He. 2023. [Language models are greedy reasoners: A systematic formal analysis of chain-of-thought](#). *Preprint*, arXiv:2210.01240.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. [Proximal policy optimization algorithms](#). *Preprint*, arXiv:1707.06347.
- 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](#). *Preprint*, arXiv:2502.21074.
- Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. 2022. [Learning to summarize from human feedback](#). *Preprint*, arXiv:2009.01325.
- DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qingqing Zheng. 2025. [Token assorted: Mixing latent and text tokens for improved language model reasoning](#). *Preprint*, arXiv:2502.03275.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Hu. 2025. [Stop overthinking: A survey on efficient reasoning for large language models](#). *Preprint*, arXiv:2503.16419.
- Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. [CommonsenseQA: A question answering challenge targeting commonsense knowledge](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.
- Minghan Wang, Thuy-Trang Vu, Ehsan Shareghi, and Gholamreza Haffari. 2025. [Towards inference-time scaling for continuous space reasoning](#). *Preprint*, arXiv:2510.12167.
- Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024. [Math-shepherd: Verify and reinforce llms step-by-step without human annotations](#). *Preprint*, arXiv:2312.08935.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-consistency improves chain of thought reasoning in language models](#). *Preprint*, arXiv:2203.11171.
- Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025. [Softcot++: Test-time scaling with soft chain-of-thought reasoning](#). *Preprint*, arXiv:2505.11484.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. [Tree of thoughts: Deliberate problem solving with large language models](#). *Preprint*, arXiv:2305.10601.
- Wengao Ye, Yan Liang, and Lianlei Shan. 2026. [Thinking on the fly: Test-time reasoning enhancement via latent thought policy optimization](#). *Preprint*, arXiv:2510.04182.
- Runyang You, Yongqi Li, Meng Liu, Wenjie Wang, Liqiang Nie, and Wenjie Li. 2026. [Parallel test-time scaling for latent reasoning models](#). *Preprint*, arXiv:2510.07745.
- Kaiyan Zhang, Jiayuan Zhang, Haoxin Li, Xuekai Zhu, Ermo Hua, Xingtai Lv, Ning Ding, Biquing Qi, and Bowen Zhou. 2025a. [OpenPRM: Building open-domain process-based reward models with preference trees](#). In *The Thirteenth International Conference on Learning Representations*.
- Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025b. [The lessons of developing process reward models in mathematical reasoning](#). *Preprint*, arXiv:2501.07301.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. [Least-to-most prompting enables complex reasoning in large language models](#). *Preprint*, arXiv:2205.10625.
- Xiangxin Zhou, Zichen Liu, Haonan Wang, Chao Du, Min Lin, Chongxuan Li, Liang Wang, and Tianyu Pang. 2025. [Variational reasoning for language models](#). *Preprint*, arXiv:2509.22637.
- 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](#). *Preprint*, arXiv:2507.06203.

Appendix

A Related Work

A.1 ITS in Discrete Space

ITS improves reasoning by allocating additional test-time compute to generate candidate solutions and select or aggregate their decoded answers. In token-based models, this sample-then-select paradigm is naturally supported by explicit next-token distributions. Self-consistency (Wang et al., 2023) samples multiple CoT solutions and aggregates them via majority voting, while best-of- N methods use likelihoods, confidence signals, or learned verifiers to rank sampled candidates (Cobbe et al., 2021; Lightman et al., 2024). pass@ k evaluation in code generation further highlights candidate-set coverage as a meaningful generation-side objective (Chen et al., 2021), and recent work directly optimizes exploration-oriented objectives such as pass@ k or entropy-driven reasoning (Chen et al., 2025; Cheng et al., 2025). Structured prompting frameworks such as Least-to-Most (Zhou et al., 2023), Tree-of-Thoughts (Yao et al., 2023), and Graph-of-Thoughts (Besta et al., 2024) further formulate reasoning as explicit search over branching intermediate states. In parallel, verifier and Process Reward Model (PRM) methods improve the selection side by evaluating final answers or intermediate reasoning steps (Wang et al., 2024; Zhang et al., 2025b,a). Together, these works show that candidate generation and answer selection are closely coupled but can be studied as distinct components of ITS. They also benefit from explicit textual trajectories and token probabilities, which make sampling, scoring, and aggregation relatively well defined in discrete space.

A.2 Continuous Space Reasoning

Continuous latent reasoning performs multi-step inference directly in hidden state space, refining latent thought representations without emitting intermediate textual tokens (Sui et al., 2025; Zhu et al., 2025). This paradigm aims to improve reasoning efficiency and representational expressivity, and CoT2 (Gozeten et al., 2025) further shows that continuous states can represent multiple reasoning traces in parallel. Existing work primarily studies how to construct, train, or organize such latent reasoning representations. COCONUT (Hao et al., 2024) executes reasoning within hidden state space; CODI (Shen et al., 2025)

aligns student and teacher hidden states through self-distillation; CCOT (Cheng and Durme, 2024) introduces variable-length latent embeddings with optional decoding; and Token Assorted (Su et al., 2025) mixes discrete and latent reasoning. Latent Thought Models (Kong et al., 2025) introduce explicit latent thought vectors with prior distributions and variational posterior inference as inference-time computation. These works demonstrate the promise of continuous thought representations, but they mainly focus on representation learning, architecture design, or training objectives. The sampling side of latent ITS remains less developed, especially for existing latent reasoning backbones whose intermediate transitions are mostly deterministic at inference time.

A.3 Continuous-Space ITS

Recent work has begun to study ITS in continuous reasoning space. SoftCoT++ (Xu et al., 2025) promotes diverse latent paths by generating multiple “soft thoughts” from distinct initial tokens, while CoT2 (Gozeten et al., 2025) represents parallel reasoning paths as superpositions of continuous tokens. Other work samples and aggregates latent trajectories more directly: Wang et al. (2025) use Monte Carlo Dropout to induce stochastic latent paths and analyze the gap between pass@ N and practical aggregation, while You et al. (2026) combine latent trajectory sampling with a learned reward model for aggregation. Probabilistic and variational formulations are also related. LaTRO (Chen et al., 2024) treats textual rationales as latent variables and optimizes them with a self-rewarding variational objective, while Variational Reasoning (Zhou et al., 2025) derives ELBO and multi-trace objectives for thinking traces and connects rejection sampling, forward-KL training, and RL-style methods such as GRPO. These works provide useful probabilistic perspectives, but mainly operate over textual traces whose likelihoods are defined by autoregressive token probabilities. Finally, LatentSeek (Li et al., 2026) and LTPO (Ye et al., 2026) optimize instance-specific latent thought vectors at test time using intrinsic signals such as model confidence, whereas our goal is to learn an amortized sampler that can be reused across inputs.

Overall, continuous-space ITS lacks the mature sampling and selection machinery available in token space. Existing methods either rely on heuristic perturbations such as dropout or fixed Gaussian noise, optimize textual reasoning traces with token-

level likelihoods, or perform per-instance latent optimization. GTS instead targets the sampling-side gap in latent ITS: it augments a frozen latent reasoning backbone with an explicit, context-conditioned Gaussian perturbation policy whose density is tractable and whose sampling behavior can be optimized with policy-gradient objectives. Thus, our contribution is not a new latent-variable language model, a verifier, or a per-instance optimizer, but a lightweight mechanism that turns heuristic latent perturbation into a learned conditional sampling policy for finite-budget latent ITS.

B Additional Details about GTS

B.1 Reward Shaping

For each input x , we sample a group of N latent perturbation trajectories and obtain N decoded answers. Let $a^{(i)}$ denote the i -th decoded answer, and let $\mathbb{I}[a^{(i)} = y^*]$ be the exact-match indicator with respect to the ground-truth answer y^* . The reward for trajectory i is defined as

$$r^{(i)} = r_0 (2 \mathbb{I}[a^{(i)} = y^*] - 1) + \alpha s^{(i)}, \quad (20)$$

where $r_0 > 0$ controls the base correctness magnitude and α scales a lightweight shaping term.

Base Correctness Term The first term assigns $+r_0$ to correct answers and $-r_0$ to incorrect ones. This symmetric formulation ensures that correctness remains the dominant optimization signal. When $\alpha = 0$, the objective reduces to accuracy-only reward.

Confidence Score For each trajectory, we compute a scalar confidence score $c^{(i)}$ using the length-normalized log-probability of the generated answer:

$$c^{(i)} = \frac{1}{|a^{(i)}|} \sum_{t=1}^{|a^{(i)}|} \log p_{\theta}(a_t^{(i)} \mid x, \tau^{(i)}, a_{<t}^{(i)}), \quad (21)$$

where $|a^{(i)}|$ denotes answer length and $\tau^{(i)}$ is the sampled latent trajectory. Length normalization prevents longer answers from being systematically penalized.

Within-Group Normalized Shaping To avoid directly optimizing raw likelihood (which could dominate correctness), we apply group-wise normalization separately over correct and incorrect

subsets. Define

$$\mathcal{C} = \{i : \mathbb{I}[a^{(i)} = y^*] = 1\}, \quad (22)$$

$$\mathcal{W} = \{i : \mathbb{I}[a^{(i)} = y^*] = 0\}. \quad (23)$$

Within each subset (when its size is at least 3), we compute a z -score normalization:

$$z(c^{(i)}) = \frac{c^{(i)} - \mu_{\mathcal{S}}}{\sigma_{\mathcal{S}}}, \quad \mathcal{S} \in \{\mathcal{C}, \mathcal{W}\}, \quad (24)$$

where $\mu_{\mathcal{S}}$ and $\sigma_{\mathcal{S}}$ are the mean and standard deviation of $c^{(i)}$ within the subset. The shaping term is then defined as

$$s^{(i)} = \begin{cases} \tanh\left(\frac{z(c^{(i)})}{\tau}\right) & i \in \mathcal{C}, |\mathcal{C}| \geq 3 \\ -\tanh\left(\frac{z(c^{(i)})}{\tau}\right) & i \in \mathcal{W}, |\mathcal{W}| \geq 3 \\ 0 & \text{otherwise.} \end{cases} \quad (25)$$

The temperature parameter τ controls the smoothness of the shaping signal, which is set as 1.0 in our experiments.

Design Rationale This construction has three desirable properties: (1) correctness remains the primary optimization objective via r_0 ; (2) shaping only provides relative discrimination *within* correct and incorrect groups, rather than encouraging unconditional likelihood maximization; and (3) the tanh squashing bounds the shaping magnitude, ensuring it remains secondary to the correctness term. Empirically, this dense signal improves within-group ranking of sampled trajectories without destabilizing training.

B.2 Implementation Details

Sampler Architecture GTS consists of two lightweight heads that predict the mean $\mu_{\phi}(c_k)$ and log standard deviation $\log \sigma_{\phi}(c_k)$ of a diagonal Gaussian policy at each latent step. Each head is implemented as a two-layer feed-forward network with hidden dimension D , SiLU activation, and output dimension D . Only the sampler parameters are trainable; all backbone parameters remain frozen.

To prevent premature collapse toward a deterministic policy, we clamp the minimum log standard deviation to -2.0 , corresponding to $\sigma_{\min} = \exp(-2) \approx 0.135$. This lower bound ensures a non-trivial exploration scale throughout training.

For COCONUT, perturbations are directly added to the latent hidden state. For CODI, which contains an additional recurrent filtering module with

layer normalization at the end, perturbations are injected *after* the recurrent filter output. This placement ensures that stochastic perturbations are not attenuated by normalization layers and can effectively propagate to subsequent reasoning steps.

In terms of parameter size, GTS introduces approximately 2.3M parameters for COCONUT (1.8% of the backbone) and 16M parameters for CODI (1.2% of the backbone).

Perturbation Schedule Across Latent Steps

Both models employ $K = 6$ latent reasoning steps. However, perturbations are injected only from `<start_latent>` through `<lat_5>`.

The final latent token `<lat_6>` influences the output only through attention to `<end_latent>` and does not re-enter the latent autoregressive loop. Specifically, the subsequent `<end_latent>` token is provided via teacher forcing when predicting the answer prefix, meaning that `<lat_6>` affects decoding through key-value cache interactions but is not recursively fed back as a new latent state. Injecting perturbations at `<lat_6>` would therefore not fully propagate through the reasoning dynamics. To ensure that stochasticity consistently influences autoregressive latent refinement, we restrict perturbations to the first five latent steps.

Policy Optimization Details The reference sampler $q_{\phi_{\text{ref}}}$ is updated as an exponential moving average (EMA) of the current policy with decay rate 0.999. The KL regularization coefficient is set to $\beta = 0.001$.

Advantages are normalized at the prompt level: for each prompt, the $N = 32$ rollout rewards are standardized before computing the policy objective. With batch size 32 prompts, each optimization step processes 32×32 sampled trajectories jointly.

When computing trajectory log densities, a diagonal Gaussian formally requires summing over all latent dimensions. In practice, directly using the unnormalized sum over high-dimensional hidden states produces very large-magnitude trajectory log densities and can make policy ratios numerically unstable. We therefore use a dimension-normalized log density when forming policy ratios:

$$\bar{\ell}_{\phi,k} = \frac{1}{D} \sum_{d=1}^D \log q_{\phi}(z_{k,d} | c_k), \quad (26)$$

and accumulate it over latent steps,

$$\bar{\ell}_{\phi}(\tau | x) = \sum_{k=1}^K \bar{\ell}_{\phi,k}, \quad (27)$$

where K denotes the perturbed latent steps. The same normalization is applied to both the current sampler and the reference sampler, so the resulting log-ratio remains a consistent scaled surrogate for comparing the two policies while keeping its magnitude suitable for optimization. The closed-form KL term is computed under the same dimension-normalized convention as the policy log-ratio, so both the policy-gradient and regularization terms are scaled consistently in high-dimensional latent space.

For the GRPO objective, we first compute the normalized trajectory log-ratio

$$\Delta \bar{\ell}^{(i)} = \bar{\ell}_{\phi}(\tau^{(i)} | x) - \bar{\ell}_{\phi_{\text{ref}}}(\tau^{(i)} | x). \quad (28)$$

For numerical stability, this log-ratio is clipped to $[-20, 20]$ before exponentiation:

$$\rho^{(i)} = \exp \left(\text{clip} \left(\Delta \bar{\ell}^{(i)}, -20, 20 \right) \right). \quad (29)$$

This clipping is applied only to the log-ratio used to compute the policy ratio and prevents overflow or underflow during exponentiation. The resulting ratio $\rho^{(i)}$ is then used in the standard clipped GRPO surrogate, where the policy ratio is clipped to $[1 - \epsilon, 1 + \epsilon]$ inside the objective, where $\epsilon = 0.2$ in our experiments. Thus, the $[-20, 20]$ clipping controls numerical stability at the log-ratio level, while the GRPO clipping controls the size of policy updates.

Training Configuration Unless otherwise stated, GTS is trained for 10K optimization steps with learning rate 1×10^{-4} and linear warmup (200 steps) with AdamW optimizer ($\beta_s = \{0.9, 0.999\}$, $\epsilon = 1e-8$, weight decay = 0). The reward shaping coefficient is $\alpha = 0.2$, $r_0 = 1.0$. All experiments are conducted on a single NVIDIA A100 GPU.

B.3 Design Choices for the Sampling Policy

Diagonal Gaussian Policy We adopt a diagonal Gaussian parameterization for the per-step perturbation distribution, which provides a practical trade-off between expressiveness, efficiency, and optimization stability. It supports dimension-wise scaling and directional steering while retaining closed-form log-density and KL divergence, whereas a full-covariance Gaussian would require $\mathcal{O}(D^2)$ parameters and introduce substantially higher memory and numerical cost in high-dimensional latent spaces. Importantly, the diagonal assumption applies only to the conditional distribution $q_{\phi}(z_k | c_k)$ at each reasoning step, not

to the full latent trajectory: previously perturbed states are propagated through the nonlinear backbone and determine subsequent conditioning contexts, so cross-dimensional dependencies can re-enter through the recurrent dynamics. We therefore view the diagonal Gaussian as a minimal tractable policy class rather than an assumption of globally independent latent dimensions; richer low-rank or full-covariance parameterizations remain possible extensions.

Additive Perturbation Formulation We model latent exploration through additive perturbations $\tilde{h}_k = h_k^{\text{det}} + z_k$. This formulation preserves the original backbone dynamics and ensures that perturbations act as local steering signals rather than replacing latent representations. Additive noise also yields a unit Jacobian transformation, allowing the perturbation density to be directly interpreted as a distribution over latent thought states. More complex transformations (e.g., multiplicative gating or learned flows) could increase flexibility but would entangle exploration with backbone dynamics and complicate policy density computation.

Relation to Dropout-Based Bayesian Sampling Dropout has been interpreted as approximate Bayesian inference, where Bernoulli masking corresponds to a variational approximation over model weights and enables predictive uncertainty estimation via Monte Carlo sampling (Gal and Ghahramani, 2016). Under this perspective, stochastic forward passes primarily serve to quantify epistemic uncertainty and improve calibration in weight space. ITS in latent reasoning, however, constitutes a search problem: the objective is to increase the probability of obtaining at least one correct reasoning trajectory under a fixed sampling budget. Accurate posterior uncertainty estimation does not necessarily imply optimal trajectory-level exploration for decision correction. Our method therefore does not aim to approximate a weight posterior, but instead learns a reward-aligned perturbation policy over latent states tailored to the inference-time objective.

C Additional Experimental Results

C.1 Out of Distribution Evaluation

To examine whether the learned exploration policy generalizes beyond the training distribution, we evaluate the samplers trained on GSM8K-Aug on three out-of-distribution arithmetic reasoning

benchmarks: MultiArith (Roy and Roth, 2015) (600 samples), SVAMP (Patel et al., 2021) (1000 samples), and GSM8K-Hard (Gao et al., 2023) (1319 samples). No additional training or hyperparameter tuning is performed for these datasets; the samplers are applied directly at inference time under the same settings used in the main experiments in Section 4.1.

Figure 5 reports the resulting pass@ N curves for both COCONUT and CODI. Across all three benchmarks, GTS consistently achieves stronger scaling behavior than heuristic baselines, including dropout-based sampling and standard Gaussian perturbations. The improvement is particularly clear at moderate-to-large sampling budgets, indicating that the learned perturbation policy transfers more consistently to unseen arithmetic benchmarks and continues to yield stronger scaling behavior than heuristic baselines.

In contrast, heuristic sampling methods exhibit noticeably inconsistent behavior across benchmarks. The same perturbation configuration can perform reasonably well on one dataset but degrade significantly on another (e.g., CODI-StandardG on MultiA and SVAMP). This variability suggests that fixed heuristic perturbations are sensitive to differences in problem difficulty, input distribution, and task characteristics. Overall, these results further support the central motivation of this work: effective sampling-side latent ITS requires better-controlled, context-aware sampling rather than globally fixed stochastic perturbations.

C.2 Sensitivity of Heuristic Perturbations

We further examine whether stronger tuning of heuristic perturbation baselines can close the gap to GTS. Table 5 reports the effect of varying the isotropic Gaussian scale σ for StandardG. The optimal scale is both backbone- and budget-dependent: COCONUT favors a relatively large scale at higher budgets, while CODI performs best around $\sigma = 0.5$ and degrades when the scale becomes too large. This confirms that fixed Gaussian perturbations require careful global scale selection and can easily move from under-exploration to disruptive perturbation.

Table 6 shows a similar sensitivity for latent dropout. For COCONUT, mild dropout with $p = 0.1$ performs best across budgets, whereas stronger dropout substantially degrades performance. For CODI, the best setting shifts to $p = 0.3$, again indicating that the appropriate perturbation strength

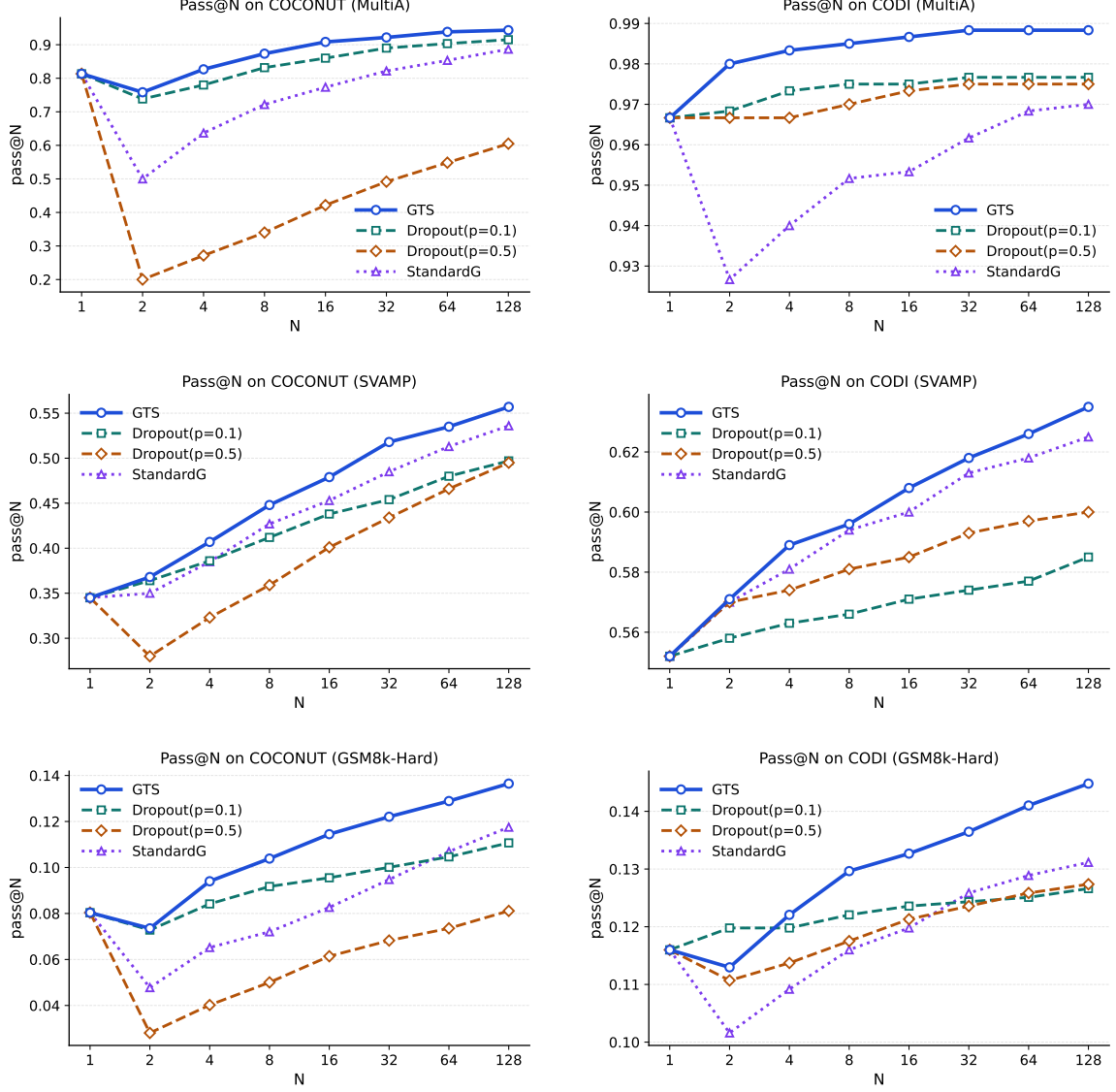


Figure 5: Out-of-distribution ITS performance on three benchmarks. Rows correspond to MultiArith, SVAMP, and GSM8K-Hard respectively, and columns correspond to COCONUT (left) and CODI (right).

depends on the latent dynamics of the backbone. These results support our main observation that heuristic perturbations can be useful, but their effectiveness is brittle because the perturbation scale is fixed globally rather than conditioned on the current latent state.

The scale sweeps leave open whether a simple state-dependent noise schedule could remove the need for such global tuning. We therefore introduce a training-free **Norm-Adaptive Gaussian** baseline (Norm-AdaptiveG), which calibrates the isotropic noise magnitude to the norm of the current latent state at every example, trajectory, and reasoning

step:

$$\mathbf{z}_k = \lambda \frac{\|\mathbf{h}_k\|_2}{\sqrt{D}} \epsilon_k, \quad \epsilon_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_D), \quad (30)$$

where D is the latent dimension and λ controls the perturbation magnitude relative to the current hidden-state norm. Under this construction, $\mathbb{E}\|\mathbf{z}_k\|_2 \approx \lambda\|\mathbf{h}_k\|_2$, so the perturbation strength adapts locally while remaining isotropic and training-free.

As shown in Table 7, norm-adaptive scaling can improve over the untuned StandardG baseline in some settings, but it does not consistently outperform a carefully chosen fixed Gaussian scale. Moreover, its best λ remains backbone- and budget-dependent. For example, COCONUT favors $\lambda =$

N	$\sigma = 0.05$	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.5$	$\sigma = 0.7$	$\sigma = 1.0$
COCONUT						
8	35.0	36.0	37.3	39.2	39.1	33.6
32	35.7	36.8	39.2	44.5	45.1	41.9
128	36.0	37.3	40.5	47.1	48.7	48.6
CODI						
8	48.6	49.4	50.1	50.8	48.9	45.2
32	48.7	50.1	51.0	52.5	51.6	48.7
128	49.1	50.5	51.8	53.6	52.0	51.1

Table 5: Standard Gaussian scale sweep on GSM8K. We report pass@ N (%) for isotropic latent perturbations with different noise scales σ . The best result for each sampling budget within each backbone is highlighted in bold.

N	$p = 0.05$	$p = 0.1$	$p = 0.3$	$p = 0.5$	$p = 0.7$	$p = 0.9$
COCONUT						
8	38.1	40.1	33.5	20.7	19.3	19.6
32	41.9	43.6	40.4	28.2	27.4	31.2
128	44.8	46.9	46.3	35.7	34.2	43.1
CODI						
8	50.0	50.1	50.8	49.4	47.5	45.7
32	50.9	51.4	52.5	51.0	49.4	49.1
128	51.3	52.6	53.5	52.2	50.9	51.8

Table 6: Effect of the dropout probability in latent dropout sampling. We report pass@ N (%) under different dropout probabilities p on GSM8K. The best result for each sampling budget within each backbone is highlighted in bold.

0.5 at smaller budgets but $\lambda = 1.0$ at $N = 128$, while CODI favors $\lambda = 0.1$ at smaller budgets and $\lambda = 1.0$ at $N = 128$. Most importantly, no tested Norm-AdaptiveG configuration matches GTS at any sampling budget on either backbone.

Together, these results show that the limitations of heuristic perturbation are not explained solely by the use of a globally fixed noise magnitude. Simple adaptation based on latent-state norm provides a stronger training-free control, but remains insufficient to reproduce the consistent scaling behavior of GTS. This suggests that GTS’s advantage cannot be reduced to state-dependent rescaling alone, and instead benefits from learning context-conditioned perturbations optimized for sampling quality.

C.3 Latent-step Robustness of GTS

We further ablate the number of latent reasoning steps used during inference. Table 8 evaluates whether GTS remains effective when the inference-time latent horizon differs from the training-time configuration. Rather than treating K as a tuned hyperparameter, this experiment tests whether the learned sampler can adapt to shorter or longer latent reasoning dynamics without collapsing under

a train-test mismatch in trajectory length. We use the same perturbation rule as in the main $K = 6$ setting: perturbations are applied to the recurrently propagated latent states, i.e., the first $K - 1$ latent positions. For $K = 1$, where no subsequent latent refinement step exists, we perturb the single available latent entry point `<start_latent>`. Thus, across all K , stochasticity is injected only where it can influence latent reasoning dynamics rather than only final answer decoding.

Across both backbones, GTS consistently improves pass@ N under unseen values of K . This suggests that the learned perturbation policy does not collapse or suffer from obvious error accumulation when the inference-time latent-step number differs from the training-time setting. Instead, GTS appears to adapt to the backbone’s latent dynamics across the tested trajectory lengths. Longer latent horizons can provide additional opportunities for useful trajectory exploration, especially on CODI; however, the gains become smaller at larger K for COCONUT, indicating a saturation effect rather than unbounded improvement from simply increasing the number of latent steps.

Method	COCONUT			CODI		
	@8	@32	@128	@8	@32	@128
Dropout ($p = 0.1$)	40.1	43.6	46.9	50.1	51.4	52.6
StandardG ($\sigma = 1.0$)	33.6	41.9	48.6	45.2	48.7	51.1
Norm-AdaptiveG ($\lambda = 0.1$)	36.2	37.0	37.6	49.5	50.1	50.6
Norm-AdaptiveG ($\lambda = 0.5$)	39.8	43.8	46.6	48.3	49.3	51.3
Norm-AdaptiveG ($\lambda = 1.0$)	34.3	41.8	47.6	46.1	49.8	52.1
GTS	42.2	49.6	55.6	51.8	55.3	57.5

Table 7: Comparison with a norm-adaptive Gaussian perturbation baseline on GSM8K. Norm-AdaptiveG scales isotropic Gaussian noise according to the norm of the current latent state, with λ controlling the relative perturbation magnitude. We report pass@ N (%). The best result for each backbone and sampling budget is highlighted in bold.

N	$K = 1$	$K = 3$	$K = 5$	$K = 6$	$K = 8$	$K = 10$
COCONUT						
1 (det.)	8.4	19.0	30.2	34.2	35.3	35.4
8	12.3	27.1	39.7	42.2	43.4	42.8
32	15.2	33.3	45.6	49.6	50.1	50.6
128	18.0	38.1	51.4	55.6	56.0	56.2
CODI						
1 (det.)	32.1	43.7	47.8	48.1	49.0	48.3
8	34.0	49.4	50.9	51.8	53.1	53.5
32	34.8	52.9	54.5	55.3	57.1	57.8
128	35.8	55.0	56.4	57.5	61.5	62.2

Table 8: Effect of the number of latent reasoning steps for GTS. We report pass@ N (%) under different latent reasoning step counts K on GSM8K. The $N = 1$ row corresponds to deterministic latent inference without stochastic sampling. The best result for each sampling budget within each backbone is highlighted in bold.

C.4 Effectiveness with Token Sampling

The main experiments use greedy answer decoding to isolate the effect of latent thought sampling. We additionally test whether GTS remains useful when combined with stochastic answer generation. As shown in Table 9, answer token sampling alone already improves over deterministic decoding, but adding latent thought sampling can further improve pass@ N . GTS combined with token sampling achieves the strongest results at moderate and large budgets on both backbones. This suggests that latent thought sampling and answer-level sampling provide complementary sources of diversity, rather than competing alternatives. It also indicates that the main greedy-decoding evaluation is a controlled setting for isolating latent exploration, not the only inference mode in which GTS can be applied.

C.5 Answer Selection

Pass@ N measures the quality of the sampled candidate set under oracle selection, but practical inference also requires selecting a final answer.

To evaluate this setting, Table 10 compares simple non-oracle selection rules, including majority voting, answer log-probability reranking, and GTS trajectory-density reranking. Across both backbones, GTS generally yields higher selected-answer accuracy than the heuristic sampling baselines under comparable selection rules. The improvement is smaller than the pass@ N gap, which is expected because answer selection remains a separate challenge from candidate generation (Wang et al., 2025).

The GTS-specific $\log q_{\text{GTS}}$ score provides a useful additional ranking signal, although it is not uniformly the best selector across all budgets and backbones. This suggests that the learned sampler captures information about trajectory plausibility, but simple density-based selection alone does not fully solve aggregation. Overall, these results support the role of GTS as a sampling-side method: it improves the candidate set and remains beneficial under basic practical selection rules, while leaving more advanced sampler-selector co-design to

N	Det.	StdG	Drop.	GTS
COCONUT				
8	42.3	36.3	44.0	43.7
32	49.6	50.6	53.0	54.9
128	56.3	57.9	59.3	64.9
CODI				
8	54.6	51.8	56.3	54.7
32	60.1	59.4	61.6	64.1
128	65.6	66.1	66.4	71.6

Table 9: Combining latent thought sampling with answer token sampling on GSM8K. We report pass@ N (%) when answer generation uses token sampling with temperature 0.7. Det., StdG, Drop., and GTS denote deterministic latent states, StandardG with $\sigma = 1.0$, latent dropout with $p = 0.1$, and the learned Gaussian thought sampler, respectively. The best result for each sampling budget within each backbone is highlighted in bold.

future work.

C.6 Additional Analysis on Steering Geometry

Finally, we analyze the geometry of the learned perturbations following identical configurations with the SNR analysis (§5.2). Table 11 reports step-wise statistics of the predicted mean perturbation μ_k and sampled perturbation z_k relative to the deterministic latent state h_k . The cosine similarity $\cos(\mu_k, h_k)$ stays close to zero for both backbones, indicating that GTS is not simply rescaling the hidden state along its radial direction. Meanwhile, the relative magnitude $\|\mu_k\|_2/\|h_k\|_2$ remains bounded across steps, despite no explicit value constraint being imposed on μ_k . This suggests that the learned mean perturbation does not exhibit runaway behavior, but instead provides stable, state-dependent steering.

The sampled perturbation magnitude $\|z_k\|_2/\|h_k\|_2$ further shows backbone-specific intervention patterns. For COCONUT, the sampled perturbation alternates between larger and smaller magnitudes across adjacent steps, while for CODI it remains more stable, with the mean perturbation becoming strongest around the middle latent step. Together with the SNR analysis (§5.2), these results suggest that GTS learns adaptive interventions aligned with the latent dynamics of each backbone, rather than applying an unconstrained or ungrounded disturbance to the latent reasoning trajectory.

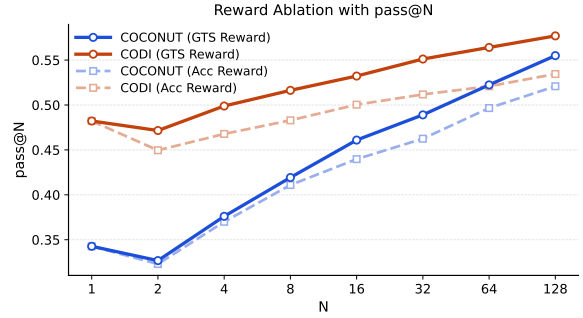


Figure 6: Ablation study on the reward shaping. Pass@ N of the GTS trained with the accuracy-only reward and the dense reward introduced in §3.3.

C.7 Stability Across Random Seeds

To examine whether the observed gains depend on a favorable sampling seed, we repeat the sampling evaluation with five random seeds {42, 43, 44, 45, 46}. As shown in Table 12, GTS achieves the highest mean pass@ N across all budgets on both backbones, while maintaining small across-seed variation. This indicates that the improvement is not driven by a single lucky rollout set, but remains stable under repeated stochastic sampling.

C.8 Ablation on Reward Shaping

We compare GTS trained with the full dense reward in Section 3.3 against an accuracy-only variant. In the simplified setting, the shaping term is removed by setting $\alpha = 0$, reducing the reward to $r^{(i)} = \{-1, 1\}$. All other training configurations remain identical. Figure 6 shows pass@ N curves on GSM8K-test for both COCONUT and CODI. The dense reward consistently outperforms the accuracy-only variant, and the performance gap widens as N increases. The shaping term effectively provides additional within-group discrimination, improving the quality of sampled candidate sets. In contrast, the accuracy-only reward treats all correct (and incorrect) samples equally, limiting refinement during training.

C.9 Computational Overhead

Table 13 reports the inference and training cost of GTS on a single NVIDIA A100-80G GPU. For inference, we profile 128 GSM8K examples with batch size 2 and $N = 32$. GTS introduces only a small runtime overhead relative to the underlying latent reasoning backbone. On COCONUT, its per-trajectory latency is comparable to dropout sampling (2.612 vs. 2.609 ms), and on CODI it is likewise comparable to StandardG (13.773 vs.

N	Dropout ($p = 0.1$)		StdG ($\sigma = 1.0$)		GTS		
	M-vote	$\log p_{\text{ans}}$	M-vote	$\log p_{\text{ans}}$	M-vote	$\log p_{\text{ans}}$	$\log q_{\text{GTS}}$
COCONUT							
8	32.1	31.5	20.4	21.0	35.1	35.2	35.8
32	32.8	32.2	21.5	22.5	36.3	35.7	36.2
128	33.0	33.4	24.1	23.3	35.9	36.8	36.1
CODI							
8	43.2	42.3	39.8	40.6	44.2	46.0	45.3
32	42.8	43.1	40.2	41.1	45.6	46.6	45.3
128	43.7	43.8	40.0	41.2	46.1	46.7	45.9

Table 10: Reranking accuracy under different latent thought sampling methods on GSM8K. We compare majority voting (M-vote), reranking by the length-normalized answer token log-probability $\log p_{\text{ans}}$, and, for GTS only, reranking by the GTS trajectory log-density $\log q_{\text{GTS}}$. The best result for each sampling budget within each backbone is highlighted in bold.

Metric	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$
COCONUT						
$\cos(\mu_k, h_k)$	0.03 ± 0.04	-0.01 ± 0.04	0.05 ± 0.04	0.01 ± 0.03	0.05 ± 0.04	-0.01 ± 0.03
$\ \mu_k\ _2 / \ h_k\ _2$	0.33 ± 0.03	0.34 ± 0.02	0.35 ± 0.03	0.34 ± 0.02	0.34 ± 0.03	0.36 ± 0.03
$\ z_k\ _2 / \ h_k\ _2$	1.01 ± 0.23	0.83 ± 0.12	1.12 ± 0.17	0.80 ± 0.14	1.10 ± 0.21	0.80 ± 0.15
CODI						
$\cos(\mu_k, h_k)$	0.05 ± 0.03	0.06 ± 0.03	-0.08 ± 0.03	-0.01 ± 0.01	-0.03 ± 0.01	0.00 ± 0.01
$\ \mu_k\ _2 / \ h_k\ _2$	0.35 ± 0.04	0.38 ± 0.05	0.58 ± 0.10	0.26 ± 0.01	0.24 ± 0.01	0.27 ± 0.03
$\ z_k\ _2 / \ h_k\ _2$	1.05 ± 0.02	1.06 ± 0.02	1.16 ± 0.06	1.04 ± 0.02	1.03 ± 0.02	1.04 ± 0.02

Table 11: Steering geometry analysis ($N = 32$) of GTS across latent reasoning steps on GSM8K. Each entry reports mean $\pm$ variance over prompts. We measure the cosine similarity between the predicted mean perturbation and the deterministic latent state, $\cos(\mu_k, h_k)$, the relative mean perturbation magnitude $\|\mu_k\|_2 / \|h_k\|_2$, and the relative sampled perturbation magnitude $\|z_k\|_2 / \|h_k\|_2$, where $z_k = \mu_k + \sigma_k \odot \epsilon_k$.

13.788 ms). Peak memory increases from 1.991 to 2.000 GB on COCONUT and from 7.059 to 7.160 GB on CODI, corresponding to less than 1.5% additional peak memory in both cases.

Sampler training is performed with batch size 8 and $N = 32$ on the same GPU. An optimization step takes 665.9 ms for COCONUT and 4368.5 ms for CODI, with peak memory usage of 6.3 GB and 11.7 GB, respectively. GTS adds approximately 2.3M parameters (1.8% of the backbone) for COCONUT and 16M (1.2%) for CODI. Since the backbone remains frozen, sampler training is a one-time cost, while the trained sampler can be reused across subsequent inference runs.

D Additional Analysis and Discussion

D.1 Further Discussion on Main Results

Beyond absolute pass@ N improvements, we observe that the *relative gains* brought by GTS differ across backbone architectures. On COCONUT, the improvement spans a wider margin (approximately $35 \rightarrow 55$), whereas on CODI the gains are more moderate (approximately $48 \rightarrow 58$). We hy-

pothesize that this discrepancy reflects structural differences in model scale and latent representation geometry.

Model Scale and Latent Dimensionality COCONUT, built on a GPT-2 backbone, operates in a lower-dimensional latent space ($D = 768$) with fewer layers. In such a regime, perturbations added to latent states can propagate more directly through subsequent reasoning steps, allowing moderate steering signals to produce visible changes in downstream predictions. By contrast, CODI employs a deeper architecture with substantially higher dimensional latent representations ($D = 2048$). In higher-dimensional spaces, meaningful directional steering becomes inherently more challenging: the action space grows with D , while the relative magnitude of any single perturbation component diminishes. Moreover, deeper networks possess stronger internal correction dynamics, which can dampen or redistribute injected perturbations across layers. As a result, while GTS remains effective on CODI, its relative gains are naturally smaller than those observed on the lower-

N	Dropout ($p = 0.1$)	Dropout ($p = 0.5$)	StdG ($\sigma = 1.0$)	GTS
COCONUT				
8	39.42 $\pm$ 0.52	20.64 $\pm$ 0.69	33.39 $\pm$ 0.39	42.09$\pm$0.24
32	43.68 $\pm$ 0.16	28.10 $\pm$ 0.55	41.94 $\pm$ 0.70	49.23$\pm$0.38
128	46.93 $\pm$ 0.49	34.77 $\pm$ 0.44	48.54 $\pm$ 0.55	55.42$\pm$0.51
CODI				
8	50.46 $\pm$ 0.14	49.48 $\pm$ 0.19	45.63 $\pm$ 0.34	52.01$\pm$0.21
32	51.45 $\pm$ 0.07	51.07 $\pm$ 0.20	48.79 $\pm$ 0.25	55.42$\pm$0.25
128	52.48 $\pm$ 0.20	52.30 $\pm$ 0.25	51.39 $\pm$ 0.43	58.02$\pm$0.41

Table 12: Stability across random seeds on GSM8K. We report pass@ N (%) as mean $\pm$ standard deviation over five random seeds {42, 43, 44, 45, 46}. The best mean result for each sampling budget within each backbone is highlighted in bold.

Method	COCONUT		CODI	
	Time	Peak mem.	Time	Peak mem.
Inference (ms/trajectory; batch size 2, $N = 32$)				
Det.	2.486	1.991 GB	13.480	7.059 GB
Dropout	2.609	1.991 GB	13.529	7.059 GB
StandardG	2.603	1.991 GB	13.788	7.059 GB
GTS	2.612	2.000 GB	13.773	7.160 GB
Sampler Training (ms/optimization step; batch size 8, $N = 32$)				
GTS	665.9	6.3 GB	4368.5	11.7 GB

Table 13: Computational overhead of GTS on GSM8K. All measurements are obtained on a single NVIDIA A100-80G GPU. Inference profiling uses 128 examples. Time is reported in milliseconds per trajectory for inference and milliseconds per optimization step for sampler training.

dimensional backbone.

Sampler Capacity Relative to Backbone Size

Although the sampler architecture adopts the same two-layer design in both settings, its *relative capacity* differs with respect to the backbone. As reported in Section B.2, GTS introduces approximately 1.8% additional parameters for COCONUT and 1.2% for CODI. While the absolute sampler size increases with latent dimension, the backbone grows more substantially in the higher-dimensional model. Consequently, the sampler-to-backbone capacity ratio becomes smaller in CODI.

Moreover, the effective control problem scales with latent dimensionality. A diagonal Gaussian policy in $D = 2048$ dimensions operates over a substantially larger action space than in $D = 768$, increasing the difficulty of learning precise context-conditioned steering directions. The same architectural design therefore faces a more demanding control landscape in larger latent manifolds.

From this perspective, the reduced relative gain on CODI does not indicate diminished effectiveness of structured perturbation, but rather re-

flects the increased complexity of steering higher-dimensional reasoning dynamics. Future work may explore model-specific sampler architectures, including deeper sampler heads or layer-wise perturbation mechanisms, to better match backbone scale. In the present work, however, we deliberately keep the sampler lightweight to isolate and validate the core idea of learnable, controlled latent exploration.

On the Small- N Behavior We also observe a mild performance drop at very small sampling budgets (e.g., $N = 2$). This phenomenon is not unique to GTS and reflects a general exploration-exploitation trade-off: when only a few samples are available, even structured perturbations may temporarily disrupt otherwise correct deterministic reasoning. As N increases, the probability that at least one trajectory meaningfully improves the internal decision state grows rapidly, leading to the observed recovery and scaling gains.

Importantly, this small- N degradation is not fundamental. One could introduce an explicit budget-aware scaling factor on the perturbation magnitude

(e.g., modulating $\mathbf{z}$ as a function of N) to suppress exploration at very small budgets. We deliberately avoid such adaptive scheduling to ensure a fair comparison across methods and sampling budgets. Designing budget-sensitive exploration control remains an interesting direction for improving ITS in continuous reasoning space.

D.2 Further Discussion on Sampling Behavior

To better understand the structural differences observed in the step-wise SNR distributions (Figure 4), we relate them to the underlying training objectives and latent reasoning formulations of the two models.

COCONUT: Paired Latent-Step Dynamics

For COCONUT, SNR exhibits a clear saw-tooth pattern across adjacent latent steps. Median values remain in a moderate range (approximately 0.3–0.5), indicating that deterministic steering and stochastic variation are comparable in magnitude throughout reasoning.

This alternating structure is consistent with COCONUT’s training design. COCONUT is trained via curriculum learning that progressively compresses textual reasoning into latent thought representations. Importantly, its original training formulation contains three effective reasoning steps, each composed of *two* latent thought vectors. Within each pair, the first sub-step primarily consolidates information, while the second refines or expands upon it.

The SNR plot suggests that GTS adapts to this internal structure. The first sub-step within each effective reasoning pair tends to exhibit relatively stronger deterministic steering, while the second allows comparatively greater stochastic exploration. Rather than imposing a rigid perturbation schedule, GTS could align with the base model’s intrinsic reasoning rhythm, preserving the alternation between consolidation and variation.

CODI: Front-Loaded Deterministic Adjustment

In contrast, CODI displays a front-loaded SNR profile. Deterministic strength peaks at the second latent step and then transitions into a sustained lower-SNR regime for subsequent steps. Later reasoning steps therefore operate in a more exploration-dominated setting.

This behavior is consistent with CODI’s training objective. Unlike COCONUT, which compresses intermediate textual reasoning through staged curriculum learning, CODI relies on distillation to

align latent reasoning with the text reasoning answer prefix at the final step. As a result, its latent trajectory is trained to evolve more smoothly and coherently across steps, without an explicit paired-step structure.

Under this formulation, a single early deterministic adjustment may be sufficient to steer the trajectory toward a promising region of latent space, after which controlled exploration can proceed without further strong intervention. The observed SNR profile therefore reflects how GTS adapts to the more continuous and globally aligned reasoning dynamics of CODI.

Adaptive Rather Than Prescriptive Control

Taken together, these patterns indicate that GTS does not enforce a uniform perturbation schedule across architectures. Instead, the balance between deterministic steering and stochastic variation emerges from interaction with each model’s latent reasoning dynamics. The SNR distributions thus suggest that context-dependent perturbations can adapt to model-specific latent dynamics, rather than uniformly amplifying or suppressing stochasticity across all reasoning steps.

D.3 A Candidate-Set View of Sampling-Side Latent ITS

We provide a simple candidate-set interpretation of sampling-side latent ITS. For an input x , let $\mathcal{T}(x)$ denote the space of latent reasoning trajectories induced by a sampler $q(\tau | x)$. Define a subset $\mathcal{U}(x) \subseteq \mathcal{T}(x)$ of *useful* trajectories, e.g., trajectories with $\Delta(\tau) > 0$ under the sampling-gain metric (defined in 4), or trajectories that decode to the correct answer. Under a sampling budget N , the probability that the candidate set contains at least one useful trajectory is

$$P_N(q; x) = 1 - (1 - q(\mathcal{U}(x) | x))^N. \quad (31)$$

This expression makes explicit that sampling-side ITS is governed by how much probability mass the sampler assigns to useful regions of trajectory space, rather than by perturbation magnitude or diversity alone. For finite N , increasing $q(\mathcal{U}(x) | x)$ directly improves the chance that the candidate set exposes a better reasoning path, while the marginal benefit naturally saturates as this probability becomes large.

This view also clarifies why distribution shift and answer diversity are insufficient proxies for sampling quality. A sampler may induce a large shift

from the deterministic trajectory by placing mass on many irrelevant or harmful regions, yielding diverse candidates but small $q(\mathcal{U}(x) \mid x)$. Conversely, a controlled perturbation may induce only moderate shift while concentrating more mass near decision-improving trajectories. Thus, effective exploration is not equivalent to moving farther in latent or answer space; it requires moving probability mass toward regions that are useful under the downstream task.

The Gaussian parameterization used by GTS can be interpreted through this lens. The learned mean provides a context-dependent directional shift that can move samples toward reusable useful regions, while the learned variance controls local exploration around that shifted center. If the variance is too small, the sampler may remain near a narrow trajectory and miss alternative useful paths; if it is too large, probability mass is diluted over irrelevant regions. A context-conditioned Gaussian therefore offers a minimal policy class for balancing directed steering and stochastic coverage under finite inference budgets. This interpretation is consistent with our empirical decomposition in Table 4, where mean-only and variance-only perturbations provide partial gains, but their combination gives the strongest scaling behavior.