How Do Latent Reasoning Methods Perform Under Weak and Strong Supervision?

Anonymous ACL submission

Abstract

Latent reasoning has recently emerged as a reasoning paradigm that performs multi-step reasoning in latent space instead of generating textual steps. While prior works mainly focus on improving performance, the internal mechanisms of latent reasoning remain not fully understood. In this work, we compare latent reasoning methods under different supervision strengths, and identify two key limitations. First, many methods exhibit shortcut behavior, achieving high accuracy without meaningfully relying on latent steps. Second, although the latent state can encode multiple possible trajectories, it does not faithfully perform the BFS-like exploration hypothesized in prior work, but instead exhibits implicit pruning and compression. Finally, we reveal a fundamental trade-off associated with supervision strength: stronger supervision mitigates shortcut behavior but limits latent exploration, whereas weaker supervision enables greater diversity at the cost of increased shortcut behavior.

1 Introduction

Latent reasoning has recently emerged as an alternative to explicit chain-of-thought reasoning, where models perform iterative multi-step reasoning in hidden space instead of explicitly generating reasoning steps in natural language (Hao et al., 2024; Deng et al., 2024; Liu et al., 2024; Shen et al., 2025). By reasoning through continuous latent representations, such approaches move beyond the limitations of discrete textual reasoning and enable more flexible forms of computation.

Despite recent development of latent reasoning algorithms, their internal mechanisms remain poorly understood. Unlike explicit reasoning, latent reasoning operates through unobservable intermediate states, making it unclear whether the algorithm truly performs multi-step reasoning. Moreover, although Hao et al. (2024) hypothesizes that latent reasoning may support BFS-like exploration by encoding multiple reasoning paths within latent representations, it remained underexplored whether this behavior actually emerges in practice.

In this work, we categorize existing latent reasoning methods into two groups based on the level of supervision imposed on latent states during training: weakly supervised methods (Hao et al., 2024; Shen et al., 2025), and strongly supervised methods (Hao et al., 2024; Shen et al., 2025). We then conduct a systematic analysis of both paradigms:

First, we investigate how latent reasoning depth affects reasoning performance. Surprisingly, we find that some methods still achieve strong performance even when latent reasoning is entirely disabled, suggesting a shortcut behavior that bypasses genuine multi-step reasoning. We then provide additional evidence of this shortcut reliance by interventional evaluations and attention-based analyses. While prior work has noted similar phenomena (Yu, 2024; Zhang et al., 2025), our analysis further reveals how different supervision schemes influence the severity of shortcut behavior.

Second, we revisit the parallel BFS hypothesis. While a single latent state can encode multiple candidate reasoning paths, we find that latent reasoning does not exactly follow BFS-style exploration. Instead, it exhibits implicit pruning and compression in latent iterations, limiting effective exploration of diverse hypotheses. These findings suggest that latent reasoning may primarily serve as a mechanism for reasoning compression, instead of realizing genuine structured search in latent space.

Finally, we reveal a trade-off governed by supervision strength: stronger supervision stabilizes reasoning and mitigates shortcuts, but limits the possibilities encoded in latent states.

Our study provides new insights into the internal mechanisms and limitations of latent reasoning models, highlighting the need for future designs of more capable and robust latent reasoning systems.

2 Background

Latent reasoning performs intermediate reasoning entirely in latent space, without explicitly decoding intermediate steps into natural language (Hao et al., 2024; Shen et al., 2025; Wei et al., 2025).

Let $x = (x_1, ..., x_n)$ denote the input tokens, $E(x)$ be the token embedding of x and F_θ be the transformer forward function. In **latent reasoning**, the model performs multi-step computation in continuous latent space by recursively reusing intermediate latent states as inputs. Let h_t be the latent state at step t. The model updates its state as:

$$e_{t+1} \leftarrow h_t, \quad h_{t+1} = F_\theta\big(E(x_1), \ldots, E(x_t), e_{t+1}\big).$$

After T steps, the model returns to text space and generates the final answer via the standard output head (with projection matrix W). Let $y = (y_1, ..., y_m)$ be the output tokens,

$$y_1 = \text{Softmax}(Wh_{t+T}), \quad y_{k+1} = \text{Softmax}(Wh_k), \text{where}$$

$$h_k = F_\theta\big(E(x_1), \ldots, E(x_t), e_{t+1} \ldots e_{t+T}, E(y_1), \ldots, E(y_k)\big).$$

According to Hao et al. (2024); Zhu et al. (2025b), as intermediate states are not collapsed into deterministic discrete tokens, a single latent can encode multiple trajectories simultaneously. Iteratively generating such latent steps has therefore been hypothesized to resemble parallel breadth-first exploration, where multiple candidate reasoning paths are implicitly maintained before final decoding.

3 Shortcut Behavior of Latent Reasoning

3.1 Experimental Settings

Latent Reasoning Methods. We evaluate four representative methods: Coconut (Hao et al., 2024), CODI (Shen et al., 2025), SIM-CoT (Wei et al., 2025) ,and CoLaR (Tan et al., 2025). Based on the training schemes, we categorize the methods into two groups: Coconut and CODI are *weakly supervised* methods, whose training rely primarily on outcome-level signals without directly constraining intermediate latent states. SIM-CoT and CoLaR belong to *strongly supervised* methods, which explicitly align latent states with intermediate reasoning steps by additional supervision. More details about the training schemes are provided in Appendix A.

Models and Benchmarks. All latent reasoning methods are trained on two backbones: GPT-2 (Radford et al., 2019) and LLaMA-3.2-1B (Dubey et al., 2024). Experiments are conducted on two datasets: GSM8K (Cobbe et al., 2021) for grade-school math reasoning, and ProsQA (Hao et al., 2024) for multi-step first-order logical reasoning. More details are provided in Appendix C.

In the experiment, we analyze how performance changes with numbers of latent steps. We train all methods using their default configurations and vary the latent lengths in inference. As CoLaR dynamically uses different latent steps across examples, we additionally enforce fixed lengths for controlled comparison. We also report results under CoLaR's default adaptive inference setting. Results are summarized in Figure 1.

3.2 Observations

If there is no shortcut reliance, we expect that (1) when the number of latents is set to zero, the prediction accuracy will be very low, and (2) when the number of latents increases, the accuracy will increase and finally saturate. However, based on Figure 1, the observations in both datasets are different from this assumption.

First, for GSM8K, when the number of latent steps increases, the accuracy in general increases for all methods. This is intuitive, as GSM8K requires multi-step numerical reasoning, and sufficient latent steps allows models to carry out more complete stepwise reasoning. However, when the latent depth is zero, accuracy does not collapse to zero. Notably, CODI retains about 30% accuracy on GSM8K with LLaMA, suggesting that a non-trivial portion of the performance arises from direct answer generation instead of stepwise reasoning.

Second, on ProsQA, most methods (except CoLaR) exhibit almost no performance change as the number of latent steps changes, even when the latent length is reduced to zero. This abnormal behavior suggests that, for ProsQA, these methods do not strongly rely on latent reasoning steps to achieve high accuracy, but capture a shortcut to answer the question. To explain this, compared to GSM8K, ProsQA involves simpler logical structures and more repetitive reasoning patterns, making it easier for models to memorize recurring reasoning patterns and rely on surface-level correlations instead of conducting genuine multi-step reasoning.

Finally, CoLaR is the most sensitive method to the latent step number. When the latent length is reduced to zero, performance on GSM8K drops to nearly zero, and on ProsQA it falls close to random-guess performance (50% for the binary task). This indicates that CoLaR relies heavily on latent states to perform reasoning. This is closely related to CoLaR's training process, which applies strong

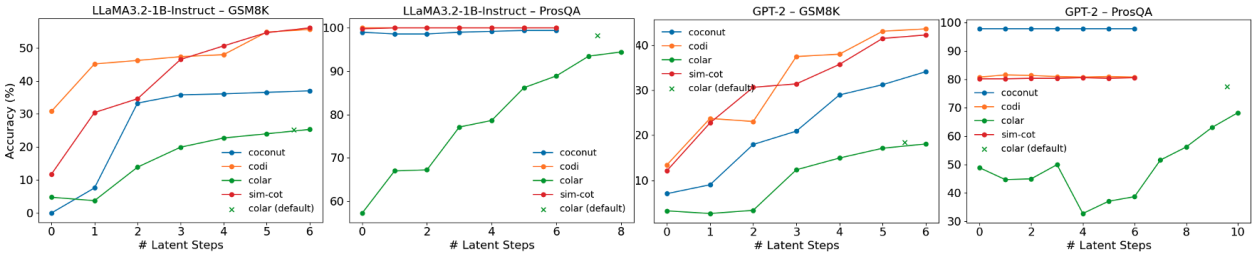

Figure 1: Performance of different methods across different numbers of latent steps

supervision by aligning latents with token-level information. With stricter learning signals beyond the final answer, the model is forced to encode meaningful information in reasoning steps, making it less likely to learn shortcut mappings.

To further diagnose shortcut behavior, we conduct additional analyses, including noise-based interventions and attention inspection in Appendix D.

Our observations suggest that shortcut naturally emerges in latent reasoning. Compared to explicit CoT, latent reasoning imposes weaker constraints on intermediate states, making it easier for models to bypass structured reasoning. The effect is more pronounced in weakly supervised methods, where learning signals are primarily outcome-level and intermediate states receive weak supervision.

4 Investigation on the BFS Mechanism

4.1 Experiment Settings

We investigate the parallel BFS hypothesis primarily on Coconut, the first latent reasoning method that explicitly motivates latent reasoning through the conjecture of BFS. To reduce the influence of shortcut behavior, we focus on GSM8K with GPT-2, which exhibits the weakest shortcut effects in Figure 1. However, in our initial experiments, we additionally observe a collapse of Coconut's reasoning pattern in inference, which prevents a systematic investigation of BFS-like exploration. We introduce the problem and the mitigation below.

Collapse of the Reasoning in Coconut. In addition to shortcut reliance, we observe a collapse behavior in Coconut when the number of latent steps at inference time is set smaller than its default configuration. Specifically, reducing the latent length causes the model to directly produce the final answer rather than generating the remaining textual reasoning steps. This deviates from the intended training design: under Coconut's stage-wise training scheme, a model using fewer latent steps should fall back to generating additional textual steps before producing the final solution. We conjecture that this collapse arises because later training stages override behaviors learned in earlier stages. Early training stages expose the model to settings with fewer latent steps and more textual reasoning, whereas later stages increasingly rely on more latent steps and fewer textual steps. As a result, the model gradually loses the ability to continue textual reasoning when latent steps are reduced at inference time.

Improved Coconut. To mitigate the issue and enable a more faithful examination of the BFS hypothesis, we modify Coconut's training scheme by allowing later stages to incorporate data from earlier stages. Specifically, at training stage k, the proportion of data sampled from stage i ($i \leq k$) is proportional to $(i+1)$, ensuring that examples with fewer latent steps remain presented throughout training. This design prevents the model from forgetting the step-wise correspondence between latent and textual steps learned in previous stages.

Empirically, under the revised training scheme, the model exhibits more stable behavior with fewer latent steps, correctly continuing the remaining reasoning instead of directly generating the final answer. Moreover, the modification also improves overall performance, even under the default fully latent inference setting. On GSM8K, GPT-2's test accuracy increases from 34.09% to 41.06%.

BFS Verification via Hybrid Rollouts. After mitigating the collapse behavior, we proceed to examine the BFS hypothesis by testing whether a latent step can encode multiple reasoning trajectories. To this end, we adopt a hybrid setup where a prefix of the reasoning process is executed in latent space, followed by explicit reasoning in text space. We fix a latent prefix and generate the remaining reasoning steps in text space using stochastic decoding with a temperature of 1. Starting with each # prefix step setting, we perform 100 independent rollouts and analyze the diversity of next-step predictions and final results. We examine how this diversity evolves as more reasoning is encoded in latent space.

To provide a controlled comparison, we construct a baseline where all reasoning steps are generated in text space. The first n reasoning steps

Table 1: Distribution of possibilities of final results with various numbers of prefix steps

# prefix step	Latent reasoning (next)				Latent reasoning (final)				Explicit reasoning (next)				Explicit reasoning (final)			
	avg	median	min	max	avg	median	min	max	avg	median	min	max	avg	median	min	max
1	18.75	14	1	87	28.35	23	1	95	3.68	2	1	87	9.32	4	1	95
2	20.38	15	1	89	25.50	19	1	96	3.31	1	1	65	6.59	2	1	99
3	20.00	13	1	97	21.82	13	1	96	2.73	1	1	80	4.17	1	1	90
4	17.22	10	1	92	17.74	10	1	94	2.01	1	1	60	2.42	1	1	70
5	15.84	9	1	91	15.84	9	1	91	1.27	1	1	27	1.37	1	1	50

Table 2: Prediction accuracy across prefix steps

	Latent reasoning		Explicit reasoning	
# prefix step	Pass@100	Maj@100	Pass@100	Maj@100
1	82.34%	44.20%	62.17%	44.12%
2	78.62%	41.70%	55.34%	44.05%
3	75.74%	39.95%	48.30%	42.71%
4	70.36%	39.73%	45.87%	43.52%
5	69.07%	39.42%	44.05%	43.59%

are decoded deterministically, while the remaining steps are generated using the same stochastic decoding procedure and number of rollouts. This design ensures that the two settings differ only in whether the initial steps are encoded in latent space or explicitly represented as text. By comparing rollout diversity, we assess whether latent states maintain a broader set of reasoning possibilities, providing empirical evidence for BFS-like behavior.

4.2 Experiment Results

According to Table 1, compared to explicit CoT, latent CoT consistently results in greater diversity in both next-step predictions and final answers across all prefix-step settings. This supports the core intuition of Coconut: by avoiding decoding intermediate steps into deterministic text, latent reasoning can encode a richer set of possible reasoning paths.

However, increasing the number of latent prefix steps does not lead to a continuous expansion of the space of the subsequent step. Instead, the number of distinct answers initially increases slightly and then consistently decreases as the latent steps increases. This suggests that latent reasoning does not exactly exhibit a BFS process. Under a true BFS mechanism, increasing the depth of latent reasoning would accumulate possibilities from earlier steps, leading to a continuous growth in the number of candidate solutions. The observed trend instead suggests that implicit pruning occurs in the latent reasoning process, limiting the diversity of surviving trajectories. This is further supported by the distribution of final outcomes, where the number of distinct solutions continues to decrease as more reasoning steps are conducted in the latent space.

To examine whether increased diversity improves accuracy, we report Pass@100 (the fraction of samples containing at least one correct solution among 100 outputs) and majority-vote accuracy in Table 2. The results show that, while latent reasoning consistently achieves higher Pass@100, it does not lead to higher accuracy under majority voting.

Therefore, although latent reasoning preserves a larger set of candidate solutions, the additional diversity does not necessarily concentrate probability mass on the correct answer. Thus, the increased outcome-level diversity alone is insufficient to guarantee effective aggregation toward the correct solution. We conjecture that this challenge arises from the limited capacity of the model to simultaneously maintain diverse candidate reasoning trajectories while reliably identifying and amplifying the correct one in the reasoning process.

To examine whether the observed trends generalize beyond Coconut, we present additional results with other latent reasoning methods in Appendix E.

Trade-off between supervision strength and latent exploration. Based on our analysis of shortcut behavior and the BFS conjecture, we identify a trade-off associated with supervision strength. While stronger supervision reduces shortcut behaviors by constraining latent representations more tightly, it limits the capacity of latent states to encode more plausible trajectories. Conversely, relaxing supervision increases representational flexibility but risks shortcut behaviors. These findings highlight the need for future work to better balance this trade-off and improve the ability to effectively explore and identify correct reasoning trajectories.

5 Conclusion

In this work, we analyze current latent reasoning models and identify several limitations. Many existing methods exhibit shortcut behavior, achieving good performance without relying on intermediate latent steps. Moreover, while latent states can encode multiple possibilities, they do not faithfully support BFS-like exploration, but exhibit implicit pruning. These findings reveal a fundamental trade-off: stronger supervision reduces shortcut behavior but also limits latent exploration, motivating future methods that better balance the two aspects.

Limitations

It is worth noting that the design of latent reasoning methods currently lacks a unified principle, and existing approaches adopt diverse training strategies. As a result, the space of possible method variants is large and still evolving. In this work, we focus on several representative methods (Coconut, CODI, SIM-CoT, and CoLaR) rather than attempting to cover the full spectrum of latent reasoning techniques.

References

Jeffrey Cheng and Benjamin Van Durme. 2024. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yuntian Deng, Yejin Choi, and Stuart Shieber. 2024. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

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 arXiv:2502.05171*.

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 arXiv:2412.06769*.

Tianqiao Liu, Zui Chen, Zitao Liu, Mi Tian, and Weiqi Luo. 2024. Expediting and elevating large language model reasoning via hidden chain-of-thought decoding. *arXiv preprint arXiv:2409.08561*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. 2025. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*.

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 arXiv:2502.21074*.

Jihoon Tack, Jack Lanchantin, Jane Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason Weston, and Xian Li. 2025. Llm pretraining with continuous concepts. *arXiv preprint arXiv:2502.08524*.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Jian Luan, and Ruihua Song. 2025. Think silently, think fast: Dynamic latent compression of llm reasoning chains. *arXiv preprint arXiv:2505.16552*.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. 2025. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. 2025. Sim-cot: Supervised implicit chain-of-thought. *arXiv preprint arXiv:2509.20317*.

Kevin Xu and Issei Sato. 2025. A formal comparison between chain-of-thought and latent thought. *arXiv preprint arXiv:2509.25239*.

Wengao Ye, Yan Liang, and Lianlei Shan. 2025. Thinking on the fly: Test-time reasoning enhancement via latent thought policy optimization. *arXiv preprint arXiv:2510.04182*.

Runyang You, Yongqi Li, Meng Liu, Wenjie Wang, Liqiang Nie, and Wenjie Li. 2025. Parallel test-time scaling for latent reasoning models. *arXiv preprint arXiv:2510.07745*.

Yijiong Yu. 2024. Do llms really think step-by-step in implicit reasoning? *arXiv preprint arXiv:2411.15862*.

Boyi Zeng, He Li, Shixiang Song, Yixuan Wang, Ziwei He, Xinbing Wang, and Zhouhan Lin. 2025. Ponderlm-2: Pretraining llm with latent thoughts in continuous space. *arXiv preprint arXiv:2509.23184*.

Yuyi Zhang, Boyu Tang, Tianjie Ju, Sufeng Duan, and Gongshen Liu. 2025. Do latent tokens think? a causal and adversarial analysis of chain-of-continuous-thought. *arXiv preprint arXiv:2512.21711*.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. 2025a. Emergence of superposition: Unveiling the training dynamics of chain of continuous thought. *arXiv preprint arXiv:2509.23365*.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart Russell, and Yuandong Tian. 2025b. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *arXiv preprint arXiv:2505.12514*.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, and 1 others. 2025c. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*.

A Methods with Weak/Strong Supervision

Based on the training objectives and supervision granularity, we categorize the existing latent reasoning methods into two groups: approaches with ***weak supervision*** and ***strong supervision***. Specifically, methods with weak supervision rely primarily on outcome-level objectives: latent reasoning is not directly constrained, but is learned only through its effects on the follow-up steps and the correctness of the final answer. In contrast, strongly supervised methods impose explicit and fine-grained supervision on latent representations, often through decoder-based objectives or token-level compression. Given this definition, we categorize the four representative methods considered in this paper:

Coconut (Hao et al., 2024) adopts a stage-wise training scheme that progressively shifts reasoning from text space to the latent space. Across training stages, the model increases the number of latent reasoning steps while correspondingly reducing the number of explicit textual reasoning steps. At each stage, supervision is provided through cross-entropy loss on the remaining textual predictions, where the model is trained to match the ground-truth in text space.

CODI (Shen et al., 2025) employs a unified training scheme without explicit stages. Its objective consists of two complementary loss terms. The first term supervises the learning of the latent steps using the final textual solution as an outcome-level objective. The second term introduces a teacher–student distillation objective, where the latent produced by a teacher model after textual reasoning are used to supervise the student's final latent states.

SIM-CoT (Wei et al., 2025) extends the latent reasoning frameworks of Coconut and CODI by introducing an additional explanation loss. Specifically, it applies the language model decoder to the intermediate latent states and optimizes a reconstruction loss that conditions on each latent state to recover the corresponding textual reasoning step, which explicitly provides stronger supervision over reasoning dynamics.

CoLaR (Tan et al., 2025) applies supervision through token-level compression. Given a compression factor c, CoLaR averages the latent representations of every c consecutive tokens and uses the resulting compressed representations as the supervision targets during training. In addition, standard supervision on the final textual answer is applied. This compression-based supervision directly constrains latent representations to align with aggregated token-level information.

Among the methods, Coconut and CODI rely on relatively weak supervision, as latent states are trained indirectly via final outputs or final latent representation alignment, whereas SIM-CoT and CoLaR impose stronger supervision through explicit alignment with intermediate textual steps or token-level representations.

B Other Related Works

Additional implicit reasoning methods In addition to the four representative methods that our analysis focuses on, there exist many other works that propose effective implicit reasoning approaches. For example, Cheng and Van Durme (2024) introduces CCoT, which utilizes continuous, variable-length contemplation tokens to conduct implicit reasoning. Liu et al. (2024) propose HCoT, which applies a two-stage framework for achieving latent representation and final answer alignment. Another stream of studies, including CoCoMix (Tack et al., 2025) and PonderLM-2 (Zeng et al., 2025), investigates pretraining strategies for continuous latent reasoning. Additionally, several recent works (Geiping et al., 2025; Saunshi et al., 2025; Wang et al., 2025; Zhu et al., 2025c) have explored recurrent or looped transformer architectures to generate continuous thoughts that simulate CoT reasoning in latent states. Other follow-up works have investigated test-time scaling approaches for latent reasoning (You et al., 2025; Ye et al., 2025).

Analytical work for latent reasoning methods Beyond proposing new methods, a few recent works have also conducted theoretical or empirical analyses to better understand the mechanisms underlying latent reasoning.

Theoretically, Zhu et al. (2025b) leverages a graph reasoning problem to prove the effectiveness of latent reasoning with a simplified transformer structure and demonstrate that each continuous thought encodes multiple searches simultaneously. Zhu et al. (2025a) further studies the training dynamics of continuous reasoning and explains how the superposition of reasoning traces

emerges in gradient-based training. Xu and Sato (2025) indicates that while latent reasoning enables efficient parallel computation in latent space, explicit reasoning supports approximate counting and sampling via stochastic decoding. Although these theoretical works provide support for the hypothesis about the parallel search in latent reasoning, their analyses are largely conducted under simplified settings, and it remains unclear to what extent the proposed parallel-search behavior emerges in practical latent reasoning systems trained on real-world data.

Empirically, Yu (2024); Zhang et al. (2025) provide analysis on whether latent reasoning generates effective intermediate reasoning steps. Yu (2024) shows that prompt-based latent reasoning often fails to induce genuine internal reasoning, while training-based approaches can produce more meaningful latent steps. Zhang et al. (2025) indicates that Coconut exhibits shortcut dependence rather than faithfully executing a reasoning process in certain datasets. While our findings partially overlap with those of these prior studies, we provide a more comprehensive analysis by systematically examining how different training schemes influence the shortcut behavior and what kind of tasks are more vulnerable to such issues.

C Additional Experimental Setup

For the experiments in Section 4, we use author-released checkpoints for CODI, SIM-CoT, and CoLaR on GSM8K whenever available. All remaining settings require training from scratch. Details of the hyperparameters can be found as follows:

- **Coconut.** For GSM8K, we use the AdamW optimizer with a learning rate of 1e-4, weight decay of 0.01, and train for 25 epochs. The maximum number of stages is set to 3, with 3 epochs per stage. For ProsQA, we use the same optimizer and learning rate, train for 50 epochs, set the maximum number of stages to 6, and use 5 epochs per stage.

- **CODI.** We use the AdamW optimizer with a learning rate of 1e-3 for GPT-2 and 2e-4 for LLaMA, weight decay 0.01, and train for 20 epochs for both models. For both models, the number of latent steps is set to 6, and LoRA is applied with rank 128 and scaling factor $\alpha = 32$.

- **SIM-CoT.** We use the AdamW optimizer with a learning rate of 1e-3 for GPT-2 and 2e-4 for LLaMA, weight decay 0.01, and train for 40 epochs for both models. For both models, the number of latent steps is set to 6, and LoRA is applied with rank 128 and scaling factor $\alpha = 32$.

- **CoLaR.** On ProsQA, we perform full fine-tuning for GPT-2 and LoRA-based fine-tuning for LLaMA (rank 128, $\alpha = 32$). For both models, we use AdamW with a learning rate of 1e-4, weight decay of 0.01, and train for 50 epochs. The maximum compression factor is set to 5.

The training sets contain 385k examples for GSM8K-Aug and 17.9k examples for ProsQA. For inference in our analysis, we use the test sets of the two datasets, which contain 1319 and 500 examples, respectively. As CoLaR supports varying compression factors, we use a default rate of 5 to maintain comparable latent step counts across methods.

Artifacts, Licensing, and Intended Use. We use publicly available datasets and pretrained models in accordance with their respective licenses and terms of use. All artifacts are used solely for research and evaluation purposes, consistent with their intended use. We do not redistribute or alter the access conditions of any third-party artifacts.

D Additional Investigation of the Shortcut Issue

D.1 Interventional Analysis

To further validate the potential shortcut behavior, we conduct an interventional analysis by applying controlled noise perturbations to the latent steps to assess the model's reliance on latent reasoning.

Specifically, after latent reasoning terminates and the model signals the transition to final answer generation, we inject a random noise sampled from a multivariate Gaussian distribution, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, into the embedding of the latent token. The model then proceeds to generate the final answer. To ensure the perturbation is strong enough to corrupt latent representations, the injected noise ($\Sigma = \sigma^2\mathbf{I}$, $\sigma = 100$) is much larger in magnitude than the original latent embeddings (the average ℓ_2 norms of the latent embeddings are 24.42 for GPT-2 and 44.08 for LLaMA3.2-1B, with embedding dimensions of 768 and 2048, respectively).

Table 3: Performance under clean inputs vs. noise perturbations for different implicit/explicit reasoning methods.

		Coconut		CODI		CoLaR		SIM-CoT		Standard CoT	
Model	Setting	GSM8K	ProsQA	GSM8K	ProsQA	GSM8K	ProsQA	GSM8K	ProsQA	GSM8K	ProsQA
LLaMA	clean	36.97%	99.40%	55.57%	100.00%	25.23%	98.20%	56.03%	100.00%	61.74%	90.60%
LLaMA	noise	20.61%	99.40%	27.45%	99.60%	3.32%	60.92%	10.08%	97.60%	0.03%	0.40%
GPT	clean	34.09%	97.80%	43.59%	80.80%	18.44%	77.52%	42.23%	80.60%	43.56%	81.00%
GPT	noise	3.79%	89.00%	8.87%	80.80%	2.84%	46.80%	7.05%	75.60%	0.08%	0.00%

Table 4: Performance different noise levels on GSM8K and ProsQA (with GPT2).

Noise level	Coconut		CODI		SIM-CoT		CoLaR ($c = 5$)	
	GSM8K	ProsQA	GSM8K	ProsQA	GSM8K	ProsQA	GSM8K	ProsQA
clean	34.09%	97.80%	43.59%	80.60%	42.23%	80.60%	18.44%	77.52%
std=100	3.79%	89.00%	8.87%	80.60%	7.05%	75.60%	2.84%	46.80%
std=10	5.68%	98.00%	12.59%	80.60%	11.60%	80.00%	1.36%	45.70%
std=1	22.65%	97.80%	41.39%	80.80%	41.93%	80.60%	8.57%	51.71%
std=0.1	33.78%	97.80%	43.44%	81.00%	42.08%	80.60%	17.82%	74.73%

Table 3 reports the accuracy of different methods before and after noise injection. For a direct comparison, we apply the same intervention to explicit reasoning by injecting noise into the embedding of the CoT token immediately before the final answer is generated (denoted as Standard CoT).

From the results in the table, we observe a clear contrast between standard CoT and latent CoT. When noise is injected into the reasoning token, the accuracy of standard CoT drops to nearly zero. In contrast, almost all latent methods retain non-zero performance under strong noise perturbations that disrupt latent representations, indicating the presence of shortcut behaviors.

The emergence of shortcut is consistent with the trends observed in Section 3. On the simpler ProsQA dataset, only CoLaR, the method with stronger supervision, shows an obvious performance degradation under noise injection, while other methods remain largely unaffected. On GSM8K, while all latent methods experience performance drops, they retain non-trivial accuracy, with the effect being most pronounced for Coconut and CODI, which rely on weaker supervision. For example, Coconut with LLaMA still achieves around 20% accuracy on GSM8K after noise, compared to a clean performance of around 34%.

These observations support our hypothesis that shortcuts naturally emerge in latent reasoning. Unlike explicit CoT, which enforces token-level supervision and tightly aligns reasoning steps to textual representations, latent reasoning lacks direct constraints on intermediate states, making it easier for models to bypass structured reasoning. Moreover, this phenomenon is particularly pronounced in methods trained with weaker supervision, where learning signals are primarily outcome-level and intermediate states receive only weak supervision.

Influence of Noise Level. We further conduct ablation studies on the influence of noise level. The results with GPT-2 are presented in Table 4.

From the results we can observe that, for settings where strong shortcut behaviors exist, the performance remains almost unchanged compared to the clean setting regardless of the injected noise magnitude. In contrast, for settings where shortcuts are weak or largely absent, the performance consistently deteriorates as the noise level increases.

Overall, these results further support our earlier conclusions regarding which settings exhibit shortcut behaviors. In particular, shortcut behaviors are more pronounced in weakly supervised latent reasoning methods and on more pattern-based reasoning tasks such as ProsQA.

D.2 Investigation with Attention Score

To further investigate shortcut behavior, we analyze token-level attention patterns during final-answer generation. Figure 2 presents an example from ProsQA using Coconut, including the question, the latent reasoning process, and the final answer (starting from ###). The final prediction, `gerpus`, consists of three tokens: `ger`, **`p`**, and `us`. For each token, we highlight the top-10 tokens with the highest attention scores using the same visual cues (underline, bold, and red, respectively).

The figure shows that the tokens with the top-10 attention scores for each answer token are exclu-

Every **z** **umpus** is a **tim**pus. Every **j** ompus is a terpus. Eva is **a** sterpus. Eva is a **z**umpus. Max is a **r** orpus. Every timpus is a lorpus. Every jompus is a rorpus. Every tumpus is a terpus. Every sterpus is a zumpus. Every terpus is a gerpus. Tom is a stor pus. Every zumpus is a jompus. Every sterpus is a timpus. Every storpus is a boompus. Tom is a boomp us. Every timpus is a tumpus. Is Eva a boompus or **ger**pus? **\n**<|start-latent|><|latent|><|latent|><|latent|><|latent|><|latent|><|latent|><|end-latent|>### Eva **is** a gerpus.

Figure 2: Tokens with Top-10 Attention Score (ProsQA)

sively drawn from the input question, rather than from latent reasoning tokens. This suggests that the final prediction primarily relies on the input question instead of the intermediate latent step.

In contrast, Figure 3 presents a GSM8K example in which, when generating the final answer 18, the top-10 attended tokens are predominantly latent tokens, indicating a substantially different pattern from that observed on ProsQA.

Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?\n<|start-latent|><|latent|><|latent|><|latent|><|latent|><|latent|><|latent|><|end-latent|> ### 18

Figure 3: Tokens with Top-10 Attention Score (GSM8K)

E BFS Analysis with Other Methods

Analysis on Methods other than Coconut. To examine whether the observed trends generalize beyond Coconut, we present additional results with other latent reasoning methods. As some methods do not support hybrid inference with partially latent and partially textual steps, we perform all reasoning in latent space and stochastically sample the final answer 100 times and analyze the resulting outcome distributions. The results with GPT-2 / LLaMA3.2-1B-Instruct are shown in Table 6.

As shown in the table, methods with weaker supervision, such as the improved Coconut, encode the largest number of distinct solution possibilities, whereas methods with stronger supervision, such as CoLaR, exhibit the smallest diversity. For example, with GPT-2, the average number of candidate final outcomes for the improved Coconut is 15.84, compared to only 3.21 for CoLaR. These observations are intuitive: methods trained with weaker or more implicit supervision allow greater freedom in intermediate reasoning, enabling the model to maintain multiple plausible solution trajectories. In contrast, methods with stronger supervision impose tighter constraints on intermediate representations, which naturally reduces solution diversity.

Notably, although CODI employs weaker supervision than CoLaR and SIM-CoT, it exhibits a smaller number of distinct solution possibilities. This might be because CODI contains stronger shortcut signals than the other two methods, biasing the model toward directly mapping inputs to final answers and thus limiting the exploration of alternative reasoning trajectories.

Additionally, with the exception of CoLaR, which combines strong supervision with limited solution diversity, all other methods achieve substantially higher Pass@100 than their greedy decoding accuracy. However, their majority-vote performance remains comparable to, or in some cases lower than, greedy decoding. For instance, with GPT-2, the Pass@100 for both SIM-CoT and CODI are around 70%, but the accuracies of the majority vote are only around 43%. This indicates that the mismatch between increased solution diversity and effective aggregation is not unique to Coconut but common in other latent reasoning methods.

Comparison between Improved Coconut and Standard Coconut.

To further demonstrate the consistency of the observed trends, we apply the same analysis to the standard Coconut method. Table 5 presents the distribution of final outcomes and corresponding accuracies under different numbers of prefix latent steps for both standard Coconut and the improved Coconut. For standard Coconut, we follow the default setting in (Hao et al., 2024) and set the total latent steps to 3. Consequently, the maximum number of prefix latent steps analyzed in Table 5 is also 3. In contrast, the improved Coconut alleviates the collapse issue and supports longer reasoning trajectories, so we extend the analysis to 5 prefix latent steps.

In terms of the distribution of the final outcome possibilities, the patterns observed in standard Coconut are generally consistent with those of the improved Coconut. In particular, the number of distinct final outcome possibilities consistently decreases as the number of prefix latent steps increases, which does not align with the expected behavior of BFS-style exploration.

From the performance perspective, when the number of prefix latent steps is smaller than the default setting, standard Coconut exhibits substantially worse performance across accuracy (greedy), majority voting and Pass@100. This degradation

Table 5: Distribution of possibilities of final results with various numbers of prefix steps

	# prefix step	Distribution of possibilities				Performance		
		avg	median	min	max	acc (greedy)	majority vote	Pass@100
Standard Coconut	1	23.60	18	1	90	14.85%	14.77%	59.39%
	2	20.61	15	1	88	24.02%	24.39%	64.09%
	3	18.45	13	1	92	34.09%	34.62%	69.17%
Improved Coconut	1	28.35	23	1	95	40.00%	82.34%	44.20%
	2	25.50	19	1	96	40.08%	78.62%	41.70%
	3	21.82	13	1	96	39.24%	75.74%	39.95%
	4	17.74	10	1	94	39.54%	70.36%	39.73%
	5	15.84	9	1	91	39.54%	69.07%	39.42%

Table 6: Distribution of possibilities and performance across methods.

Model	Method	Distribution of possibilities				Performance		
		avg	median	min	max	acc (greedy)	majority vote	Pass@100
GPT-2	Improved Coconut	15.84	9	1	91	39.54%	39.42%	69.07%
	CODI	12.96	6	1	83	43.59%	42.23%	70.43%
	SIM-CoT	13.57	7	1	89	42.23%	43.21%	69.60%
	CoLaR	3.21	1	1	45	18.44%	18.42%	23.28%
LLaMA3.2-1B	Improved Coconut	10.00	5	1	84	39.68%	40.21%	59.00%
	CODI	6.39	2	1	67	55.41%	55.57%	73.84%
	SIM-CoT	7.46	2	1	65	56.01%	55.50%	72.93%
	CoLaR	7.63	2	1	73	25.48%	25.70%	33.21%

Table 7: Performance under clean and noisy latent for different variants of Coconut.

	Coconut	+ Cross-stage data reuse	+ Next-step-only supervision	+ Both
Clean	97.80%	97.40%	93.60%	93.60%
Noise	97.80%	94.80%	93.80%	4.40%

is caused by the collapse behavior discussed earlier. Specifically, when the number of latent steps used at inference time is lower than the default training configuration, the model often fails to continue reasoning in the textual space after the latent reasoning phase and instead directly outputs the final answer prematurely, resulting in insufficient reasoning steps and degraded performance.

F Effects of Improved Coconut in Mitigating Shortcut

In this section, we provide some empirical results showing that the improved Coconut training scheme we introduced in Section 4 can help mitigate the shortcut issue of Coconut in the ProsQA dataset.

Notably, in addition to the improved training scheme, which reuses data from the previous stages in the later stages, an additional modification of the training scheme is required to mitigate the shortcut issue in datasets with structured reasoning, like ProsQA. Specifically, in the early training stages, when conditioning on previous latent steps, supervision is applied only to the next-step prediction, rather than to the remaining reasoning steps or the final answer. The intuition behind this design is to avoid consistently exposing the final solution during training, especially in the later stages of the training, thereby preventing the model from learning a direct question-to-answer mapping and encouraging stepwise reasoning.

To demonstrate the mitigation of shortcut behavior, Table 7 reports performance under latent noise injection, together with ablation results for different variants of the proposed design choices.

As shown in the table, when both components, including the cross-stage data reuse and next-step-only supervision in early training stages, are applied, model performance drops to around 4% under latent noise perturbations. This sharp degradation indicates that shortcut reliance has been effectively mitigated, as disrupting latent representations now severely impairs performance. In contrast, when only one of the two components is applied, the model retains relatively high accuracy under noise, suggesting that shortcut behavior is still present. These results indicate that both design choices are necessary to effectively eliminate shortcut reliance in Coconut.