

SpiralThinker: Latent Reasoning through an Iterative Process with Text–Latent Interleaving

Shengmin Piao
Yonsei University
Seoul, South Korea
shengminp@yonsei.ac.kr

Sanghyun Park[†]
Yonsei University
Seoul, South Korea
sanghyun@yonsei.ac.kr

Abstract

Recent advances in large reasoning models have been driven by reinforcement learning and test-time scaling, accompanied by growing interest in latent rather than purely textual reasoning. However, existing latent reasoning methods lack mechanisms to ensure stable reasoning dynamics in latent space and a systematic way to interleave implicit and explicit reasoning. We introduce SpiralThinker, a stabilized iterative latent reasoning framework that performs iterative updates over latent representations while interleaving latent and textual reasoning steps. At its core, it combines a progressive alignment objective that explicitly regulates latent representations across iterations with structured annotations for text–latent interleaving, thereby stabilizing latent updates and maintaining coherence with textual reasoning. Across mathematical, logical, and common-sense reasoning tasks, SpiralThinker achieves state-of-the-art performance among latent reasoning baselines. Further analysis shows that both iteration and alignment are essential, that the optimal numbers of latent tokens and iterations vary by dataset, and that proper alignment is crucial for effective iterative latent reasoning. Overall, SpiralThinker bridges iterative computation and latent reasoning, demonstrating that aligned iterative updates can reliably steer reasoning in the latent space. Code is available at <https://github.com/shengminp/SpiralThinker>.

1 Introduction

Recent advances in large reasoning models have been driven primarily by progress in reinforcement learning (Guo et al., 2025) and test-time scaling (Snell et al., 2025), which have improved models’ reasoning capability to generate longer and more intricate chains of thought (Wei et al., 2022). In parallel, another line of research explores latent reasoning, where reasoning unfolds within high-

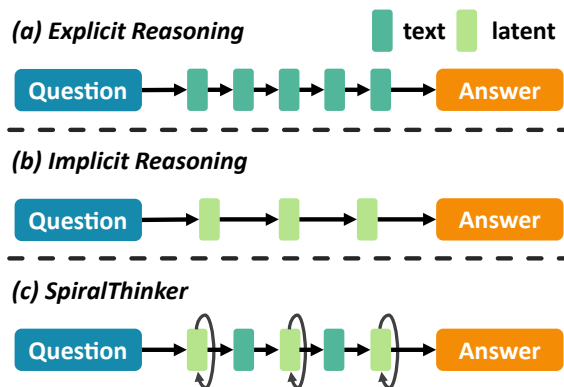


Figure 1: (a) Explicit reasoning processes textual tokens once. (b) Implicit reasoning processes latent representations once. (c) SpiralThinker interleaves textual with latent reasoning via an iterative process.

dimensional hidden representations rather than explicit textual sequences (Hao et al., 2024). Despite their apparent differences, both paradigms share the same objective: enriching a model’s internal computation, either explicitly through text tokens or implicitly through hidden representations¹.

Existing latent reasoning methods show promise in modeling the underlying reasoning trajectory (Chen et al., 2025a; Zhu et al., 2025; Li et al., 2025) but remain underexplored in two critical respects.

One underexplored aspect concerns the mechanisms for performing stable and coherent reasoning in latent space. Most approaches treat latent representations as token-level inputs (Goyal et al., 2024; Hao et al., 2024; Shen et al., 2025) and process them in a single forward pass (Zhu et al., 2025). However, this token-centric perspective forces latent representations to encode entire reasoning steps simultaneously, redistributing computation without modeling how reasoning unfolds

¹Throughout this paper, latent (or implicit) reasoning refers to reasoning that occurs within hidden representations, whereas textual (or explicit) reasoning denotes reasoning expressed through textual tokens.

[†] Corresponding author.

in latent space. We argue that latent reasoning should be explicitly modeled as an iterative process, wherein latent representations are progressively enhanced to support subsequent reasoning and the final answer.

Another unexplored aspect involves how to interleave explicit and implicit reasoning at the step level. Human reasoning naturally alternates between internal thinking and external expression, whereas textual-only frameworks verbalize every step, risking either overthinking (Chen et al., 2025b) or underthinking (Wang et al., 2025). Latent approaches, in contrast, compress the reasoning steps into latent representations (Deng et al., 2023; Hao et al., 2024; Shen et al., 2025), thereby sacrificing interpretability and controllability. Bridging these paradigms requires a method capable of transitioning between interpretable textual reasoning and compact latent reasoning.

To address these gaps, we propose *SpiralThinker* (Figure 1), a stabilized iterative latent reasoning framework that performs implicit reasoning through iterative refinement of latent representations while systematically integrating it with explicit reasoning. At each iteration, *SpiralThinker* updates latent representations to deepen the reasoning trajectory without generating additional tokens, enabling extended implicit reasoning within a standard autoregressive decoding process.

To ensure coherent and goal-directed latent reasoning, we introduce a progressive alignment objective that constrains latent representations to remain consistent with their explicit textual counterparts throughout the iterative process. This objective stabilizes iterative updates and promotes coherent information flow throughout the iterative process. Furthermore, *SpiralThinker* employs a structured annotation scheme that formalizes text–latent interleaving as an alternating reasoning process, guiding the model on when to alternate between explicit and implicit reasoning steps. Together, these mechanisms yield deeper yet more coherent latent reasoning trajectories, effectively bridging explicit interpretability with implicit computational depth.

We evaluate *SpiralThinker* on three datasets: GSM8K-Aug (Deng et al., 2023), ProQA (Hao et al., 2024), and StrategyQA (Geva et al., 2021), which cover mathematical, logical, and common-sense reasoning tasks. Experimental results show that *SpiralThinker* consistently outperforms existing latent reasoning methods and adapts robustly to diverse reasoning tasks.

Empirically, ablation studies on the iterative process and the alignment objective highlight their complementary roles in enabling stable iterative reasoning. We further analyze the impact of the number of latent tokens, iterations, and the weighting schedule, revealing dataset-specific optima for the first two and confirming that appropriate alignment is essential for effective latent reasoning with the iterative process.

The key contributions of this work are as follows:

- We introduce *SpiralThinker*, a stabilized iterative latent reasoning framework that explicitly regulates latent updates across iterative refinement steps and formalizes structured text–latent interleaving within a unified reasoning process.
- We design a progressive alignment objective that provides iteration-aware supervision for latent representations across refinement steps, maintaining coherence with the corresponding textual reasoning and ensuring stable, goal-directed latent updates.
- We identify dataset-specific optima for the number of latent tokens and iterations, and demonstrate that proper alignment is crucial for achieving effective latent reasoning with the iterative process.

2 Related Work

2.1 Latent Reasoning

In the current paradigm dominated by decoder-only architectures, latent reasoning primarily manifests in the model’s internal representations—namely token embeddings, hidden states, and output logits—serving as the substrates where reasoning unfolds implicitly.

Early attempts introduced a latent token represented by a learnable embedding that is excluded from the language modeling objective. This design preserves the semantics of the original vocabulary while delaying output generation to extend internal computation (Goyal et al., 2024; Herel and Mikolov, 2024; Pfau et al., 2024). Subsequent works treated the embeddings associated with reasoning steps as continuous information carriers, applying compression techniques to obtain compact representations that can be integrated into the language modeling objective (Cheng and Van Durme, 2024; Su et al., 2025; Xu et al., 2025).

Shifting focus from token embeddings to the model’s internal hidden states, researchers explored how to provide effective learning signals. Some studies employed curriculum learning to gradually internalize reasoning capabilities (Deng et al., 2024; Liu et al., 2024), while others aligned hidden states of latent tokens with corresponding explicit reasoning steps, thereby enabling the model to process latent tokens as internal analogues of textual reasoning (Deng et al., 2023; Shen et al., 2025).

Extending this idea to the logit level, researchers have observed that a model’s output distributions already encode partial reasoning information (Wang and Zhou, 2024). This insight motivated approaches that either reuse logits directly as next-token proxies (Hao et al., 2024) or use their probability weights to blend token embeddings (Zhang et al., 2025), both aiming to establish a continuous information flow by feeding output logits back as input.

However, previous methods either treat latent representations as isolated tokens processed in a single forward pass or compress explicit reasoning into static approximations. In contrast, our approach models multiple latent representations as an integrated reasoning step and applies iterative computation to achieve deeper and more coherent latent reasoning.

2.2 Iterative Reasoning

Iterative computation, the idea of refining representations through repeated updates, is a long-standing concept in machine learning and has been widely employed in architectures such as recurrent neural networks (Goodfellow et al., 2016). Building on the Universal Transformer (Dehghani et al., 2019), which demonstrated that incorporating recurrence into the Transformer architecture (Vaswani et al., 2017) enhances performance on both algorithmic and language understanding tasks, subsequent research has further shown that iterative computation facilitates implicit reasoning.

Recent studies have explored various ways to integrate recurrence while preserving the Transformer architecture, emphasizing iterative processing with text tokens rather than latent representations. Some approaches treat the entire model as an iterative unit that refines hidden states across multiple iterations, thereby enabling implicit reasoning within hidden states (Saunshi et al., 2025). Other methods introduce block-level recurrence, reusing the same block during both training and

inference to balance computational cost and reasoning accuracy (Geiping et al., 2025; Yu et al., 2025). A complementary line of work applies recurrence selectively within layers, assigning additional iterative updates to difficult tokens without increasing the overall parameter count (Chen et al., 2025c).

Overall, recurrence represents more than additional computation, as it naturally aligns with stepwise reasoning. Theoretically, an iterative model with T iterations can simulate a T -step reasoning process, establishing a one-to-one correspondence between the iterative process and the reasoning steps (Saunshi et al., 2025).

However, existing iterative approaches typically rely solely on the standard language modeling objective, which offers no direct supervision over the latent reasoning dynamics. In contrast, our work introduces a progressive alignment objective that explicitly guides and stabilizes latent reasoning throughout the iterative process.

3 Methodology

This section presents the overall framework of SpiralThinker (Figure 2). We begin with the explicit reasoning stage (§3.1), which establishes stepwise reasoning capability, and then detail the implicit reasoning stage, consisting of the iterative process (§3.2.2) and the latent adapter (§3.2.3). Next, we describe the alignment and total training objectives (§3.2.4), followed by the inference strategy (§3.3), which explains how SpiralThinker alternates between explicit and implicit reasoning during generation.

3.1 Explicit Reasoning

We begin by fine-tuning the model under an explicit reasoning setting, which equips it with fundamental step-by-step reasoning capability and lays the foundation for the subsequent implicit reasoning stage.

Formally, given a question Q , the target sequence R comprises two components: (i) a sequence of reasoning steps and (ii) the final answer. During training, the model learns to generate both elements conditioned on Q . The training objective is defined as:

$$\mathcal{L}_{\text{CE}} = -\log P(R | Q) \quad (1)$$

Each reasoning step is enclosed by boundary tokens <bot> (begin-of-text) and <eot> (end-of-text), which later facilitate alignment with latent representations.

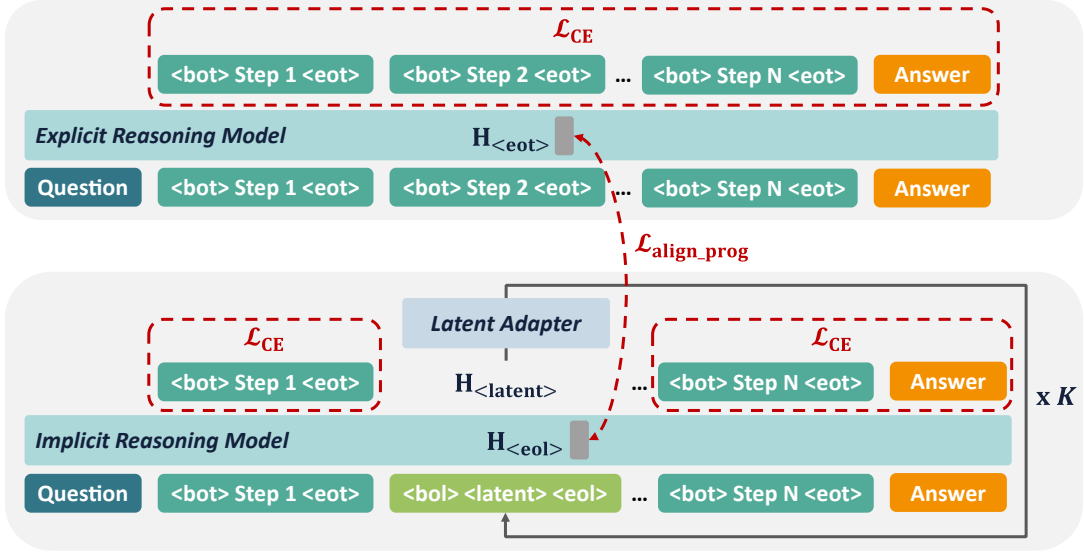


Figure 2: Training process of SpiralThinker. Step indicates a textual step, and <latent> indicates a latent step. Only one <latent> token is illustrated for clarity.

3.2 Implicit Reasoning

3.2.1 Data Construction

To enable the model to transition between textual and latent reasoning within a unified framework, we construct an alternating text–latent scheme. Specifically, every textual reasoning step at odd or even positions is replaced by N latent tokens <latent>, forming a latent step while keeping the final answer unchanged. This scheme allows textual and latent reasoning to coexist within the same sequence, enabling the model to learn transitions between the two modes during training.

As with textual steps, each latent step is delimited by <bol> (begin-of-latent) and <eol> (end-of-latent). The complete data format is described in Appendix A.

3.2.2 Iterative Process

Given the constructed scheme, we denote the input as $\mathbf{x} = [x_1, \dots, x_T]$, where each token x_t represents either a textual or latent token. Each token is projected into the embedding space through an embedding matrix $E \in \mathbb{R}^{V \times d}$:

$$\mathbf{e}_t = E[x_t] \quad (2)$$

where V denotes the vocabulary size and d the hidden dimension. The resulting embedding sequence $\mathbf{E} = [\mathbf{e}_1, \dots, \mathbf{e}_T]$ is then passed through a stack of L layers:

$$\mathbf{H}^{(l)} = f^{(l)}(\mathbf{H}^{(l-1)}), \quad \mathbf{H}^{(0)} = \mathbf{E} \quad (3)$$

yielding final hidden states $\mathbf{H}^{(L)} = [\mathbf{h}_1^{(L)}, \dots, \mathbf{h}_T^{(L)}]$.

We then perform iterative updates over latent representations. At iteration k ($k = 1, \dots, K$), we extract the hidden states corresponding to latent tokens, denoted as $\mathbf{H}_{\langle \text{latent} \rangle}^{(L, k-1)}$, and transform them using a mapping module $g_\phi(\cdot)$:

$$\tilde{\mathbf{H}}_{\langle \text{latent} \rangle}^{(k)} = g_\phi\left(\mathbf{H}_{\langle \text{latent} \rangle}^{(L, k-1)}\right) \quad (4)$$

The transformed states are then written back into their corresponding positions in the embedding sequence to form an updated input $\mathbf{E}^{(k)}$. A subsequent forward pass conditioned on $\mathbf{E}^{(k)}$ produces updated hidden states $\mathbf{H}^{(L, k)}$. Repeating this procedure for K iterations enables continuous reasoning over latent representations and fosters coherent integration of information between latent and contextual representations.

3.2.3 Latent Adapter

Since final-layer hidden states and input embeddings reside in distinct subspaces (Zhang et al., 2025), we implement $g_\phi(\cdot)$ as a lightweight adapter that aligns hidden states with the embedding scale. This adapter projects latent representations back into the embedding space before rewrite, ensuring compatibility between latent and textual representations.

Formally, the adapter comprises a residual multi-layer perceptron followed by RMSNorm and scal-

ing:

$$\tilde{\mathbf{h}} = \text{norm}(\mathbf{h} + W_2 \text{SiLU}(W_1 \mathbf{h})) \cdot \text{target_rms} \quad (5)$$

where $W_1, W_2 \in \mathbb{R}^{d \times d}$ are learnable parameters, and $\text{SiLU}(\cdot)$ denotes the SiLU activation function.

The fixed scalar factor `target_rms` is derived from the pretrained embedding matrix, excluding the newly added special tokens, as

$$\text{target_rms} = \mathbb{E}_{i \in V_{\text{base}}} \left[\sqrt{\frac{1}{d} \sum_{j=1}^d E_{ij}^2} \right] \quad (6)$$

where V_{base} denotes the pretrained vocabulary size. This ensures that the mapped latent representations remain consistent with the base embedding distribution.

3.2.4 Alignment Objective

Although the iterative process enables repeated updates of latent representations, it does not inherently guarantee that these representations encode meaningful reasoning trajectories. We therefore introduce an alignment objective that constrains each latent representation to align with its corresponding textual counterpart.

Intra-iteration alignment For each training instance, we perform a forward pass to obtain hidden states at `<eol>`, and use a frozen model trained under the explicit reasoning setting (§3.1) to extract the corresponding `<eot>` hidden states for each textual reasoning step. Since the implicit model is initialized from the same explicitly trained backbone, latent and textual representations start from a shared reasoning-aware representation space, which helps reduce distributional mismatch during implicit reasoning training. These tokens occur immediately after latent and textual reasoning steps, respectively, and serve as compact summaries of their preceding reasoning content. Importantly, the aligned hidden states are not isolated step representations: they are prefix-conditioned states of the underlying language model, encoding cumulative information from all preceding reasoning steps.

Formally, let $\mathbf{H}_{\langle \text{eol} \rangle}^{(l)}$ and $\mathbf{H}_{\langle \text{eot} \rangle}^{(l)}$ denote the layer- l hidden states at `<eol>` and `<eot>`, respectively. The alignment objective is defined as

$$\mathcal{L}_{\text{align}} = \frac{1}{L} \sum_{l=1}^L \frac{\|\mathbf{H}_{\langle \text{eol} \rangle}^{(l)} - \mathbf{H}_{\langle \text{eot} \rangle}^{(l)}\|_1}{\sigma^{(l)}} \quad (7)$$

where $\sigma^{(l)}$ is a per-layer normalization factor estimated from the frozen model as the average feature-wise standard deviation of hidden states across textual steps. This normalization maintains scale consistency across layers and stabilizes optimization (Shen et al., 2025).

By aligning hidden states at these positions, the objective provides indirect supervision for latent tokens. Although these tokens are not directly optimized, the alignment loss propagates informative gradients through their hidden states, encouraging them to encode reasoning information consistent with explicit reasoning steps while operating over a continuously evolving reasoning state rather than enforcing strict equivalence on isolated steps.

Progressive alignment While the intra-iteration alignment objective (7) aligns representations within a single pass, the iterative process introduces an additional temporal dimension across multiple iterations. During this process, earlier iterations are expected to explore diverse reasoning trajectories, whereas later iterations should progressively consolidate these into accurate reasoning steps. To capture this progression, we compute the alignment loss at each iteration and aggregate them using a weighting schedule that emphasizes later iterations.

Specifically, we define a softmax-based weighting vector that increases monotonically with the iteration index:

$$\mathbf{v} = \text{softmax}(\alpha[1, \dots, K]), \quad \alpha > 0 \quad (8)$$

where α is adaptively scaled with the number of iterations, assigning greater weights to later iterations. This weighting schedule increases the contribution of later iterations, encouraging the model to prioritize more consolidated reasoning states. At the same time, it preserves signals from earlier exploratory passes, thereby providing trajectory-level guidance rather than enforcing strict step-wise equivalence across iterations. The progressive alignment objective is therefore defined as

$$\mathcal{L}_{\text{align_prog}} = \sum_{k=1}^K v_k \mathcal{L}_{\text{align}}^{(k)} \quad (9)$$

Total objective Integrating all components described above, we formulate the final training objective as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{align_prog}} \quad (10)$$

Methods	GSM8K-Aug (%)	ProsQA (%)	StrategyQA (%)
iCoT-KD (Deng et al., 2023)	24.11	98.00	62.88
iCoT-SI (Deng et al., 2024)	29.72	<u>99.00</u>	60.69
Token Assorted (Su et al., 2025) ^a	48.70	96.20	-
Coconut (Hao et al., 2024)	49.85	97.80	60.00
CODI (Shen et al., 2025)	51.02	80.80	60.70
Pause Token (Goyal et al., 2024)	<u>53.37</u>	95.80	57.64
SpiralThinker	56.56	99.40	63.32

^a As the official code is not publicly released, the baseline results are taken from the papers.

Table 1: Accuracy (%) of SpiralThinker compared with baselines. **Bold** numbers indicate the best performance, and underlined numbers indicate the second-best.

where λ controls the weight of the progressive alignment term. Because the `<latent>` token has no explicit surface form, its prediction is excluded from the supervision signal in \mathcal{L}_{CE} .

3.3 Inference Strategy

During inference, we prepend N `<latent>` tokens to each input question. The model first performs implicit reasoning over these latent tokens, and subsequently produces explicit textual reasoning steps conditioned on the updated latent representations. Owing to the alternating training scheme, it learns to autonomously insert N `<latent>` tokens after each textual step, thereby alternating between implicit and explicit reasoning throughout the generation process. This interleaving continues until the model generates the final answer or reaches the predefined generation limit. All results are obtained using greedy decoding to ensure deterministic reasoning trajectories.

4 Experimental Setup

4.1 Dataset & Metrics

We evaluate SpiralThinker on three reasoning benchmarks—GSM8K-Aug, ProsQA, and StrategyQA—which target mathematical, logical, and commonsense reasoning, respectively. These datasets collectively span diverse reasoning domains, requiring both reasoning capability and relevant knowledge to achieve competitive performance. Detailed dataset descriptions are provided in Appendix B.

Performance is measured using answer-level accuracy, defined as the proportion of instances whose final predicted answer exactly matches the ground-truth label across all benchmarks.

4.2 Baselines

We compare SpiralThinker with recent latent reasoning approaches, retraining all publicly available implementations under identical model architectures and datasets to ensure a fair comparison. Comprehensive descriptions of all baselines are given in Appendix C.

4.3 Implementation Details

We adopt Llama-3.2-1B (Grattafiori et al., 2024) as the backbone and apply LoRA fine-tuning (Hu et al., 2022) to both SpiralThinker and all baselines. Training is conducted on four A100 GPUs using the Hugging Face library (Wolf et al., 2019), with initialization from publicly released pretrained weights². Complete hyperparameter configurations are provided in Appendix E.

5 Results and Analysis

5.1 Overall Performance

To evaluate the overall reasoning capability of SpiralThinker, we compare it with baselines on mathematical, logical, and commonsense reasoning benchmarks. As shown in Table 1, SpiralThinker consistently outperforms prior methods across all tasks. On GSM8K-Aug, it achieves 56.56%, surpassing the previous best by 3.19 points, indicating that the iterative process particularly enhances multi-step numerical reasoning. On ProsQA, it reaches 99.40%, surpassing the previous best by 0.40 points and suggesting better stability in high-accuracy regimes. On StrategyQA, the score of 63.32% exceeds the previous best by 0.44 points, reflecting stronger commonsense reasoning.

Taken together, these cross-benchmark improvements suggest that modeling implicit reasoning

²<https://huggingface.co/meta-llama/Llama-3.2-1B>

Alignment	Iteration	GSM8K-Aug (%)	ProsQA (%)	StrategyQA (%)
✗	✗	45.49	98.00	59.39
✓	✗	48.67 (+3.18)	98.60 (+0.60)	61.14 (+1.75)
✗	✓	45.72 (+0.23)	97.40 (-0.60)	58.08 (-1.31)
✓	✓	56.56 (+11.07)	99.40 (+1.40)	63.32 (+3.93)

Table 2: Ablation study on the effect of applying the iterative process and alignment objective.

as an iterative process over latent representations yields generalizable gains beyond specific tasks.

5.2 Ablation Study

To disentangle the contributions of SpiralThinker’s two core components—the iterative process and the progressive alignment objective—we conducted a series of ablations. We began with a single-iteration baseline employing latent tokens but trained only with the cross-entropy objective. We then added the alignment objective to encourage meaningful latent representations. Next, we incorporated the iterative process while retaining only the cross-entropy objective, isolating the effect of iteration. Finally, we combined both components to evaluate their joint effect.

As shown in Table 2, employing the iterative process alone yields limited improvement and can even degrade performance, suggesting that iterative updates in latent representation space may drift when left unconstrained. In contrast, incorporating the alignment objective—either in the single-pass or iterative setting—consistently enhances results, indicating that structured guidance is important for stabilizing latent reasoning. When both mechanisms are combined, the gains become substantial, confirming that iteration and progressive alignment play complementary roles: iteration provides additional reasoning depth, while progressive alignment regulates this multi-step evolution to keep it structured, stable, and coherent.

5.3 Analysis Study

We further examine the influence of three key factors: the number of latent tokens, the number of iterations, and the weighting schedule of the progressive alignment objective on the performance of SpiralThinker. This analysis provides deeper insights into the model’s sensitivity and internal dynamics.

The effect of number of latent tokens We first examined how varying the number of latent tokens ($N \in [1, 10]$) influences overall performance. To

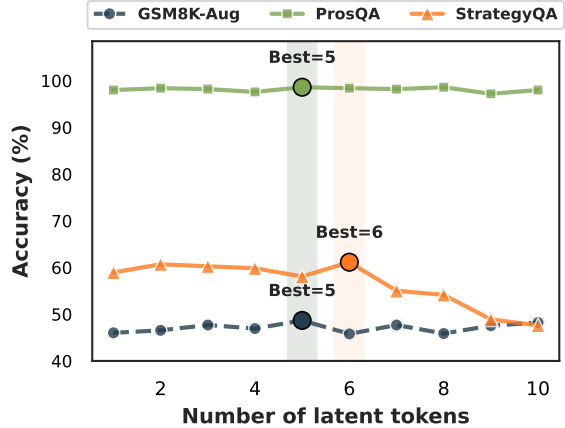


Figure 3: Accuracy on different datasets as the number of latent tokens varies.

control for confounding factors, this experiment excluded iteration while retaining the alignment objective.

As shown in Figure 3, performance on StrategyQA decreases when $N > 6$, whereas accuracy on other datasets remains largely stable across the range $N \in [1, 10]$. This trend is consistent with the observations reported by Goyal et al. (2024) and Shen et al. (2025), indicating that the optimal number of latent tokens depends on the dataset and reflects inherent differences in reasoning complexity.

Overall, the best results are obtained with $N=5$ on GSM8K-Aug and ProsQA, and with $N=6$ on StrategyQA; these configurations are used in other experiments.

The effect of number of iterations Next, we analyze how the number of iterations ($K \in [1, 6]$) in the iterative process affects performance.

As illustrated in Figure 4, a moderate number of iterations consistently improves accuracy, while excessive iteration leads to saturation or slight degradation—most notably on GSM8K-Aug and ProsQA. Although StrategyQA exhibits minor fluctuations, it follows a similar trend, suggesting that increasing iterations does not necessarily enhance

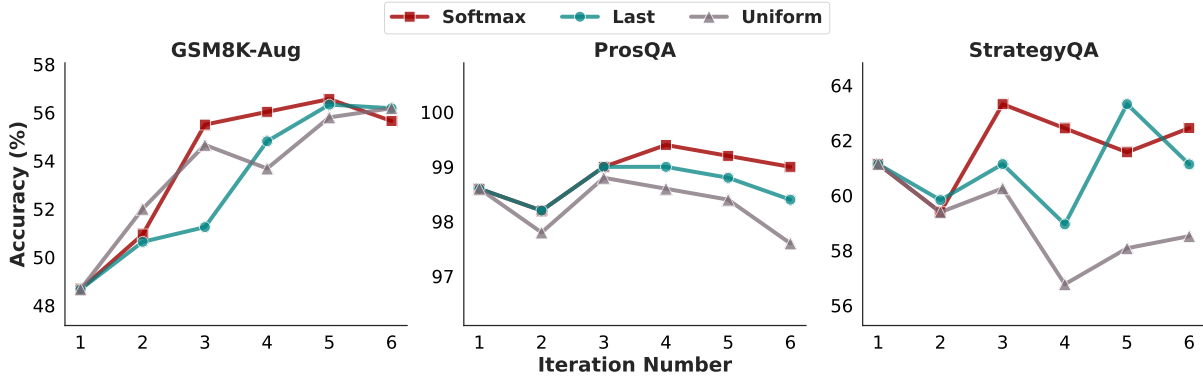


Figure 4: Accuracy on different datasets as the number of iterations varies.

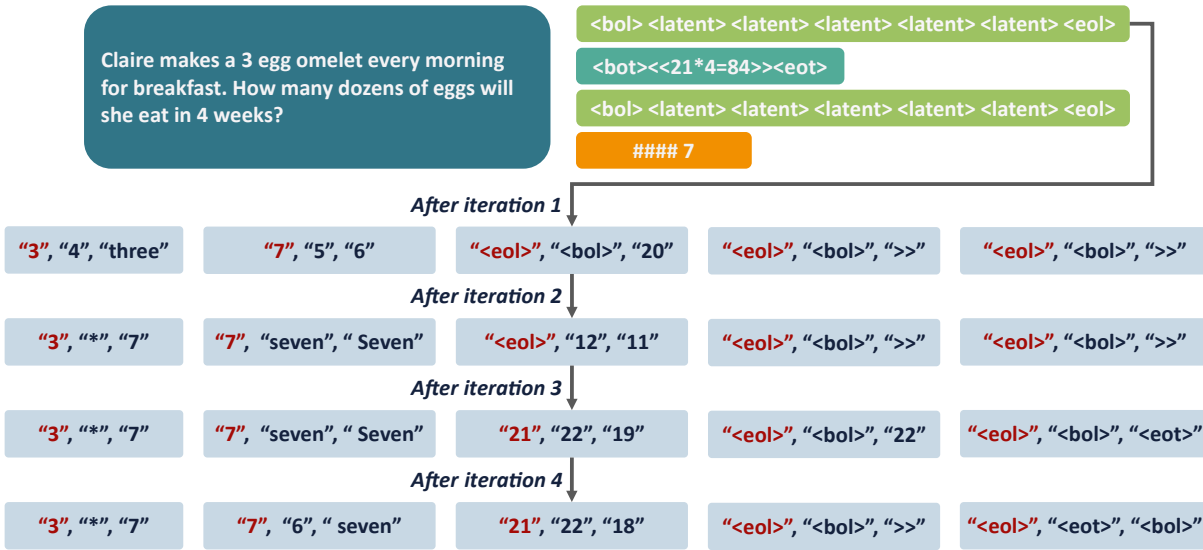


Figure 5: The upper part shows the reasoning steps generated by SpiralThinker for a sample problem, while the lower part presents the top three tokens most similar to each latent representation at the first latent step during the iterative process. The top-ranked token is highlighted in red.

reasoning performance.

Accordingly, when using the softmax-based weighting schedule, the best results are achieved with $K=5$ on GSM8K-Aug, $K=4$ on ProsQA, and $K=3$ on StrategyQA, again suggesting that the optimal number of iterations is dataset-dependent. These settings are adopted in other experiments.

The effect of the weighting schedule Finally, we investigated the effect of different weighting schedules for the progressive alignment objective. Specifically, we compared (1) uniform weighting across iterations, (2) weighting only the last iteration, and (3) our proposed softmax-based scheduler.

As illustrated in Figure 4, uniform weighting consistently underperforms, with the effect most pronounced on ProsQA and StrategyQA. In contrast, both last-iteration-only weighting and the

softmax-based weighting schedule improve performance, with the latter generally performing best: it achieves the highest peak accuracy on GSM8K-Aug and ProsQA, and converges to its performance plateau with fewer iterations on StrategyQA. These findings highlight that maintaining appropriately weighted alignment across iterations is crucial for effective latent reasoning during the iterative process.

5.4 Qualitative Analysis of `<latent>`

To assess how SpiralThinker utilizes latent representations during the iterative process, we conduct a qualitative analysis of the `<latent>` tokens. Since the latent adapter aligns latent representations with token embeddings, we identify tokens whose embeddings are most similar to these latent representations. Using a GSM8K-Aug test

instance as a case study, we report the top three tokens associated with each latent representation across iterations.

As shown in Figure 5, the first and second latent tokens consistently capture relevant reasoning factors throughout iterations. Notably, the first latent token encodes the multiplication operator “*”, indicating sensitivity to arithmetic operators. The third latent token primarily stores intermediate results, which progressively converge to the correct values. The fourth and fifth latent tokens frequently produce the <eol> symbol, marking reasoning termination. Collectively, these observations indicate that the latent representations appear to capture reasoning-relevant signals that evolve coherently throughout the iterative process.

6 Conclusion

We presented SpiralThinker, a framework that bridges iterative computation and latent reasoning through repeated updates over latent representations. By incorporating a progressive alignment objective and structured annotations, SpiralThinker maintains consistency between latent and textual reasoning, enabling extended implicit reasoning without generating additional tokens. Experiments on GSM8K-Aug, ProsQA, and StrategyQA demonstrate that SpiralThinker consistently surpasses previous latent reasoning approaches. Ablation results further confirm that both iteration and alignment are essential for stable and coherent reasoning, and appropriate alignment proves critical for an effective iterative process. In conclusion, SpiralThinker bridges iterative computation and latent reasoning, demonstrating that aligned iterative updates can reliably steer latent reasoning.

Limitations

While SpiralThinker achieves competitive performance, there remain two main areas for improvement.

First, SpiralThinker currently applies a fixed number of iterations across all reasoning steps, regardless of their difficulty. A more desirable approach would dynamically adjust the number of iterations based on instance difficulty—using fewer iterations for easy steps and more for challenging ones. In practice, this sensitivity could potentially be mitigated through lightweight calibration on a small held-out set or adaptive stopping criteria based on convergence or confidence signals.

Second, although the proposed design enables interleaved reasoning between textual and latent representations through its data scheme and training strategy, such interleaving may not be necessary at every reasoning step. Future work could investigate adaptive mechanisms that allow the model to decide when latent reasoning should be activated.

Acknowledgments

This work was supported by the National Research Foundation (NRF) funded by the Korean Government (MSIT) (No. RS-2025-00523472); by an IITP grant funded by the Korean Government (MSIT) (No. RS-2020-II201361, Artificial Intelligence Graduate School Program (Yonsei University)); and by the Regional Innovation System & Education (RISE) program through the Jeju RISE Center, funded by the Ministry of Education (MOE) and the Jeju Special Self-Governing Province, Republic of Korea (2025-RISE-17-003).

References

- Xinghao Chen, Anhao Zhao, Heming Xia, Xuan Lu, Hanlin Wang, Yanjun Chen, Wei Zhang, Jian Wang, Wenjie Li, and Xiaoyu Shen. 2025a. [Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning](#). *arXiv preprint arXiv:2505.16782*.
- Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025b. [Do NOT think that much for 2+3=? on the overthinking of long reasoning models](#). In *Forty-second International Conference on Machine Learning*.
- Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. 2025c. [Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28241–28259, Vienna, Austria. Association for Computational Linguistics.
- 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*.

- Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. [Universal transformers](#). In *International Conference on Learning Representations*.
- 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*.
- Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. 2023. [Implicit chain of thought reasoning via knowledge distillation](#). *arXiv preprint arXiv:2311.01460*.
- Jonas Geiping, Sean Michael 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](#). In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models*.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. [Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies](#). *Transactions of the Association for Computational Linguistics*, 9:346–361.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT press Cambridge.
- Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2024. [Think before you speak: Training language models with pause tokens](#). In *The Twelfth International Conference on Learning Representations*.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. [The llama 3 herd of models](#). *arXiv preprint arXiv:2407.21783*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shitong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#). *arXiv preprint arXiv:2501.12948*.
- Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. 2024. [Training large language model to reason in a continuous latent space](#).
- David Herel and Tomas Mikolov. 2024. [Thinking tokens for language modeling](#). *arXiv preprint arXiv:2405.08644*.
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- Jindong Li, Yali Fu, Li Fan, Jiahong Liu, Yao Shu, Chengwei Qin, Menglin Yang, Irwin King, and Rex Ying. 2025. [Implicit reasoning in large language models: A comprehensive survey](#). *arXiv preprint arXiv:2509.02350*.
- Tengxiao Liu, Qipeng Guo, Xiangkun Hu, Cheng Jiayang, Yue Zhang, Xipeng Qiu, and Zheng Zhang. 2024. [Can language models learn to skip steps?](#) *Advances in Neural Information Processing Systems*, 37:45359–45385.
- Jacob Pfau, William Merrill, and Samuel R. Bowman. 2024. [Let’s think dot by dot: Hidden computation in transformer language models](#). In *First Conference on Language Modeling*.
- Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. 2025. [Reasoning with latent thoughts: On the power of looped transformers](#). In *The Thirteenth International Conference on Learning Representations*.
- 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*.
- Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. [Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning](#). In *The Thirteenth International Conference on Learning Representations*.
- DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. 2025. [Token assorted: Mixing latent and text tokens for improved language model reasoning](#). In *Forty-second International Conference on Machine Learning*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). *Advances in neural information processing systems*, 30.
- Xuezhi Wang and Denny Zhou. 2024. [Chain-of-thought reasoning without prompting](#). *Advances in Neural Information Processing Systems*, 37:66383–66409.
- Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, and 1 others. 2025. [Thoughts are all over the place: On the underthinking of o1-like llms](#). *arXiv preprint arXiv:2501.18585*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). *Advances in neural information processing systems*, 35:24824–24837.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and 1 others. 2019. [Huggingface’s transformers: State-of-the-art natural language processing](#). *arXiv preprint arXiv:1910.03771*.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025. [SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23336–23351, Vienna, Austria. Association for Computational Linguistics.

Qifan Yu, Zhenyu He, Sijie Li, Xun Zhou, Jun Zhang, Jingjing Xu, and Di He. 2025. [Enhancing auto-regressive chain-of-thought through loop-aligned reasoning](#). *arXiv preprint arXiv:2502.08482*.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. 2025. [Soft thinking: Unlocking the reasoning potential of llms in continuous concept space](#). *arXiv preprint arXiv:2505.15778*.

Rui-Jie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, and 1 others. 2025. [A survey on latent reasoning](#). *arXiv preprint arXiv:2507.06203*.

A Data Format

DATA FORMAT	
Explicit Reasoning:	<pre> Question <bot> Step 1 <eot> <bot> Step 2 <eot> <bot> Step 3 <eot> <bot> Step 4 <eot> #### Answer </pre>
Implicit Reasoning:	<pre> (1) Question <bol> N x <latent> <eol> <bot> Step 2 <eot> <bol> N x <latent> <eol> <bot> Step 4 <eot> #### Answer (2) Question <bot> Step 1 <eot> <bol> N x <latent> <eol> <bot> Step 3 <eot> <bol> N x <latent> <eol> #### Answer </pre>

Table 3: Data formats used for explicit and implicit reasoning.

Table 3 summarizes the data formats used for explicit and implicit reasoning.

B Datasets

GSM8K-Aug GSM8K (Cobbe et al., 2021) is a benchmark of grade-school math problems widely used to evaluate multi-step numerical reasoning involving both language understanding and mathematical computation. GSM8K-Aug (Deng et al., 2023) is an augmented version generated with GPT-4 based on the original GSM8K training set.

ProsQA ProsQA (Hao et al., 2024) is a synthetic question–answering dataset designed to evaluate logical reasoning capability. It is constructed from randomly generated directed acyclic graphs that specify the known conditions and reasoning dependencies. Each instance requires multi-step planning to identify the correct reasoning chain, thereby assessing the model’s capability to perform structured logical reasoning.

StrategyQA StrategyQA (Geva et al., 2021) is a binary (yes/no) question-answering dataset used to evaluate commonsense reasoning across diverse topics. It challenges models to perform multi-step reasoning by decomposing complex questions into intermediate steps before producing the final answer.

Since the original dataset lacks textual reasoning steps, we use Llama-3.2-7B (Grattafiori et al., 2024) with a few-shot prompting setup (Table 8) following chain-of-thought (Wei et al., 2022) to generate them. Moreover, because the official test set is unavailable, we follow prior work by using the validation set for testing and sampling an equal-sized subset from the training data as the new validation set.

The dataset statistics are summarized in Table 4.

C Baselines

Pause Token (Goyal et al., 2024) This method introduces pause training, which allows a language model to perform additional computation before generating the reasoning process. During both training and inference, special <pause> tokens are inserted to delay token generation, enabling the model to refine its hidden representations prior to producing the next token.

ICoT-KD (Deng et al., 2023) This approach proposes implicit chain-of-thought reasoning, in which reasoning occurs within hidden states rather than through explicit text. It comprises three stages: (1) a student model learns from a teacher’s hidden

Dataset	# Train (explicit)	# Train (implicit)	# Validation	# Test
GSM8K-Aug	384,620	643,762	500	1,319
ProsQA	17,886	35,772	300	500
StrategyQA	1,714	3,412	229	229

Table 4: Dataset statistics.

Models	GSM8K-Aug (%)	ProsQA (%)	StrategyQA (%)
Llama-3.2-1B	56.56	99.40	63.32
Qwen2.5-0.5B	46.85	96.80	58.95
Qwen2.5-1.5B	58.45	99.40	65.07

Table 5: Accuracy (%) of SpiralThinker across backbone families and model sizes.

states during intermediate reasoning; (2) an emulator predicts the teacher’s hidden states directly from inputs through layer-wise reasoning; and (3) the emulator and student are jointly trained to produce final answers without explicit reasoning steps.

ICoT-SI (Deng et al., 2024) This method introduces a curriculum-based framework for transforming explicit reasoning into implicit reasoning. Starting with a model trained via explicit chain-of-thought supervision, it gradually removes intermediate steps during fine-tuning, encouraging the model to internalize reasoning within its hidden states.

Coconut (Hao et al., 2024) This approach enables reasoning in a continuous latent space rather than through discrete text tokens. Instead of decoding output logits into tokens, the model iteratively feeds the final hidden state back as the next input embedding. Training follows a multi-stage curriculum that progressively replaces explicit reasoning steps with continuous latent representations.

Token Assorted (Su et al., 2025) This hybrid framework integrates latent and textual reasoning tokens to enhance efficiency. A VQ-VAE encoder-decoder compresses the initial portion of a reasoning step into discrete latent tokens while retaining later steps as text, thereby reducing sequence length and computational cost.

CODI (Shen et al., 2025) This framework compresses explicit reasoning into a compact continuous latent representation. CODI jointly trains a teacher model with textual reasoning process and a student model using latent tokens, aligning their hidden states via a layer-wise loss. This self-distillation mechanism effectively transfers reason-

ing capability from the language space to the continuous latent space.

D Additional Experiments

D.1 Method Scalability

We assess the scalability of SpiralThinker along two dimensions: backbone family and model size. To examine cross-family transferability, we replace the original Llama backbone with Qwen2.5, another decoder-only Transformer family. To study scaling with respect to model size, we further evaluate SpiralThinker on two Qwen2.5 variants, namely Qwen2.5-0.5B and Qwen2.5-1.5B.

As shown in Table 5, SpiralThinker can be effectively trained on both the Llama-3.2 and Qwen2.5 families, suggesting that the method is not restricted to a specific backbone and can extend to different decoder-only Transformer models. Moreover, within the Qwen2.5 family, performance consistently improves as the model size increases from 0.5B to 1.5B, yielding gains of 11.60 points on GSM8K-Aug, 2.60 points on ProsQA, and 6.12 points on StrategyQA. These results suggest that SpiralThinker exhibits favorable scalability across both model families and model sizes.

D.2 Computational Cost Analysis

To better understand the computational implications of SpiralThinker, we compare it with the explicit reasoning model (§3.1) on all benchmarks using the same Llama backbone. SpiralThinker is not designed as a reasoning acceleration method. Although latent reasoning can reduce the need to externalize reasoning as textual tokens, each latent reasoning step incurs additional forward passes. Consequently, lower visible verbosity does not necessarily imply lower inference cost.

Benchmark	Method	Acc.	Avg. textual tokens	Text reduction	Avg. latent steps	Avg. latent iters	Total forward count
GSM8K-Aug	Explicit	62.02%	30.56	–	0.00	0	30.56
	SpiralThinker	56.56%	13.99	54.22%	1.71	5	22.54
ProsQA	Explicit	97.60%	44.71	–	0.00	0	44.71
	SpiralThinker	99.40%	33.57	24.92%	2.83	4	44.89
StrategyQA	Explicit	64.19%	77.14	–	0.00	0	77.14
	SpiralThinker	63.32%	39.77	48.44%	2.13	3	46.16

Table 6: Comparison between explicit reasoning and SpiralThinker on computation cost.

Table 6 reports both task performance and computation-related statistics. **Acc.** denotes task accuracy. **Avg. textual tokens** is the average number of textual tokens generated during inference. **Text reduction** measures the relative reduction in generated textual tokens for SpiralThinker compared with the explicit reasoning baseline. **Avg. latent steps** denotes the average number of latent reasoning steps generated by SpiralThinker. **Avg. latent iters** denotes the predefined number of iterative refinements applied to each latent reasoning step for a given benchmark.

To reflect the effective inference cost, we report the **Total forward count**. For the explicit reasoning model, generating one token corresponds to one forward pass; therefore, the total forward count is equal to the number of generated textual tokens. For SpiralThinker, the total forward count includes both the forward passes used to generate textual tokens and the additional forward passes introduced by the iterative process:

$$F_{\text{total}} = T_{\text{text}} + S_{\text{latent}} \cdot I_{\text{latent}}. \quad (11)$$

Here, F_{total} denotes the total forward count, T_{text} the average number of generated textual tokens, S_{latent} the average number of latent steps, and I_{latent} the average number of latent iterations.

As shown in Table 6, SpiralThinker substantially reduces the number of generated textual tokens on all three benchmarks, with reductions of 54.22% on GSM8K-Aug, 24.92% on ProsQA, and 48.44% on StrategyQA. However, its effect on total computation varies across datasets because latent reasoning introduces additional forward passes. On GSM8K, SpiralThinker reduces the total forward count from 30.56 to 22.54, indicating that the reduction in textual generation outweighs the cost of latent refinement. A similar trend is observed on StrategyQA, where the total forward count decreases from 77.14 to 46.16. In contrast, on ProsQA, although SpiralThinker generates fewer textual tokens, the additional cost of latent reasoning results in a nearly

identical total forward count (44.89 vs. 44.71).

These results suggest that SpiralThinker should be viewed as a method that offers a controllable trade-off between computation and reasoning quality, rather than primarily optimized for efficiency. Its main advantage lies in reducing textual verbosity while preserving competitive reasoning performance, whereas the actual computational cost depends on the balance between fewer generated text tokens and the extra forward passes introduced by latent reasoning.

E Hyperparameter Settings

The hyperparameters used in this study are summarized in Table 7. Because the StrategyQA dataset is relatively small compared with the other datasets, slightly different hyperparameter settings were applied.

F Generation Results

We report results on the GSM8K-Aug, ProsQA, and StrategyQA datasets, with two representative examples for each shown in Tables 9, 10, and 11, respectively, to qualitatively illustrate SpiralThinker’s reasoning behavior across different domains.

Hyperparameter	Value	Note
<i>Explicit Reasoning</i>		
epochs	5/15	Number of training epochs for GSM8K-Aug and ProsQA/StrategyQA.
batch_size	128	Batch size for GSM8K-Aug, ProsQA and StrategyQA.
lr	1×10^{-4}	Learning rate for the AdamW optimizer.
<i>Implicit Reasoning</i>		
epochs	5/30	Number of training epochs for GSM8K-Aug and ProsQA/StrategyQA.
batch_size	128/32	Batch size for GSM8K-Aug and ProsQA/StrategyQA.
lr	5×10^{-5}	Learning rate for the AdamW optimizer.
	0.8	Proportion of total weight assigned to the later iteration.
λ	0.5	Weight coefficient for the alignment objective term.
<i>LoRA Configuration</i>		
r	32/16	Rank for GSM8K-Aug and ProsQA/StrategyQA.
lora_α	64/32	Scaling factor for GSM8K-Aug and ProsQA/StrategyQA.
lora_dropout	0.05	Dropout probability applied to LoRA layers.

Table 7: Summary of hyperparameter settings used in the experiments.

PROMPT FOR STRATEGYQA

1. Do hamsters provide food for any animals?

Hamsters are prey animals.

Prey are food for predators.

Thus, hamsters provide food for some animals.

So the answer is yes.

2. Could Brooke Shields succeed at University of Pennsylvania?

Brooke Shields went to Princeton University.

Princeton University is about as academically rigorous as the University of Pennsylvania.

Thus, Brooke Shields could also succeed at the University of Pennsylvania.

So the answer is yes.

3. Hydrogen's atomic number squared exceeds number of Spice Girls?

Hydrogen has an atomic number of 1.

1 squared is 1.

There are 5 Spice Girls.

Thus, Hydrogen's atomic number squared is less than 5.

So the answer is no.

4. Is it common to see frost during some college commencements?

College commencement ceremonies can happen in December, May, and June.

December is in the winter, so there can be frost.

Thus, there could be frost at some commencements.

So the answer is yes.

5. Could a llama birth twice during War in Vietnam (1945-46)?

The War in Vietnam was 6 months.

The gestation period for a llama is 11 months, which is more than 6 months.

Thus, a llama could not give birth twice during the War in Vietnam.

So the answer is no.

6. Would a pear sink in water?

The density of a pear is about $0.6g/cm^3$, which is less than water.

Objects less dense than water float.

Thus, a pear would float.
So the answer is no.

Table 8: The prompt for StrategyQA dataset.

RESULT OF GSM8K-AUG

1. Toulouse has twice as many sheep as Charleston. Charleston has 4 times as many sheep as Seattle. How many sheep do Toulouse, Charleston, and Seattle have together if Seattle has 20 sheep?

SpiralThinker:

```
<bol><latent><latent><latent><latent><latent><eol>  
<bot><<2*80=160>><eot>  
<bol><latent><latent><latent><latent><latent><eol>  
#### 260
```

Ground Truth:

```
<<20*4=80>>  
<<2*80=160>>  
<<20+160+80=260>>  
#### 260
```

2. Claire makes a 3 egg omelet every morning for breakfast. How many dozens of eggs will she eat in 4 weeks?

SpiralThinker:

```
<bol><latent><latent><latent><latent><latent><eol>  
<bot><<21*4=84>><eot>  
<bol><latent><latent><latent><latent><latent><eol>  
#### 7
```

Ground Truth:

```
<<3*7=21>>  
<<4*21=84>>  
<<84/12=7>>  
#### 7
```

Table 9: Generated results of GSM8K-Aug.

RESULT OF PROSQA

1. Every kerpup is a sterpup. Every vumpup is a gerpup. Rex is a impup. Rex is a vumpup. Every boompup is a terpup. Every shumpup is a zhorpup. Alex is a kerpup. Every terpup is a felpup. Bob is a zhorpup. Every fompup is a gerpup. Every yimpup is a jelpup. Every gwompup is a sterpup. Every gwompup is a zhorpup. Every yimpup is a kerpup. Alex is a gwompup. Every chorpup is a terpup. Every vumpup is a lempup. Every vumpup is a shumpup. Every shumpup is a boompup. Rex is a chorpup. Every impup is a fompup. Every chorpup is a impup. Every lempup is a boompup. Every vumpup is a fompup. Alex is a jelpup. Every jelpup is a scrompup. Every shumpup is a lempup. Every impup is a shumpup. Every chorpup is a zhorpup. Alex is a tumpup. Every gwompup is a yimpup. Alex is a yimpup. Is Rex a felpup or sterpup?

SpiralThinker:

```
<bol><latent><latent><latent><latent><latent><eol>
```

<bot>Every chorpus is a terpus.<eot>
<bol><latent><latent><latent><latent><latent><eol>
Rex is a felpus.

Ground Truth:

Rex is a chorpus.
Every chorpus is a terpus.
Every terpus is a felpus.
Rex is a felpus.

2. Every impus is a lempus. Every timpus is a jompus. Every terpus is a shumpus. Every sterpus is a zhorpus. Every vumpus is a zhorpus. Eva is a timpus. Every rompus is a yumpus. Every yumpus is a lempus. Every yumpus is a impus. Every gorpus is a zhorpus. Every fompus is a quimpus. Rex is a vumpus. Every fompus is a zhorpus. Every zhorpus is a zumpus. Every sterpus is a yerpus. Every timpus is a impus. Every rompus is a timpus. Every lempus is a chorpus. Every vumpus is a fompus. Every zhorpus is a terpus. Every yerpus is a gwompus. Eva is a rompus. Every vumpus is a gorpus. Every vumpus is a sterpus. Every vumpus is a gwompus. Eva is a lempus. Is Rex a shumpus or jompus?

SpiralThinker:

<bol><latent><latent><latent><latent><latent><eol>
<bot>Every vumpus is a zhorpus.<eot>
<bol><latent><latent><latent><latent><latent><eol>
<bot>Every terpus is a shumpus.<eot>
Rex is a shumpus.

Ground Truth:

Rex is a vumpus.
Every vumpus is a zhorpus.
Every zhorpus is a terpus.
Every terpus is a shumpus.
Rex is a shumpus.

Table 10: Generated results of ProsQA.

RESULT OF STRATEGYQA

1. Is Miami a city on the American West Coast?

SpiralThinker:

<bol><latent><latent><latent><latent><latent><latent><eol>
<bot>The American West Coast refers to the western coast of the United States, which includes states like California, Oregon, and Washington.<eot>
<bol><latent><latent><latent><latent><latent><latent><eol>
no

Ground Truth:

Miami is located in the state of Florida.
Florida is on the East Coast of the United States.
Thus, Miami is not on the West Coast.
no

2. Can amoebas get cancer?

SpiralThinker:

<bol><latent><latent><latent><latent><latent><latent><eol>

<bot>Cancer is a disease that arises from abnormal cell growth and division.<eot>
<bol><latent><latent><latent><latent><latent><latent><eol>

<bot>Thus, amoebas cannot get cancer.<eot>

no

Ground Truth:

Cancer happens when cells in a multicellular organism grow uncontrollably and harm the body.

Amoebas are single-celled organisms, so they don't have tissues or organs that cancer can affect.

Thus, amoebas cannot get cancer in the way animals or humans do.

no

Table 11: Generated results of StrategyQA.