Efficient Post-Training Refinement of Latent Reasoning in Large Language Models

Anonymous ACL submission

Abstract

Reasoning is a key component of language understanding in Large Language Models. While Chain-of-Thought prompting enhances performance via explicit intermediate steps, it suffers from sufficient token overhead and a fixed reasoning trajectory, preventing step-wise refinement. Recent advances in latent reasoning address these limitations by refining internal reasoning processes directly in the model's latent space, without producing explicit outputs. However, a key challenge remains: how to effectively update reasoning embeddings during post-training to guide the model toward more accurate solutions. To overcome this challenge, we propose a lightweight post-training framework that refines latent reasoning trajectories using two novel strategies: 1) Contrastive reasoning feedback, which compares reasoning embeddings against strong and weak baselines to infer effective update directions via embedding enhancement; 2) Residual embedding refinement, which stabilizes updates by progressively integrating current and historical gradients, enabling fast yet controlled convergence. Extensive experiments and case studies are conducted on five reasoning benchmarks to demonstrate the effectiveness of the proposed framework. Notably, a +5% accuracy gain on MathQA without additional training. Code and data are publicly available at this link.

1 Introduction

Reasoning serves as a fundamental capability in Large Language Models (LLMs), enabling them to comprehend prompts and effectively solve complex tasks. Existing approaches, such as Chain-of-Thought (CoT) (Wei et al., 2022b) and ReAct (Yao et al., 2023b), guide models toward correct answers by explicitly generating intermediate textual reasoning steps. While these methods have shown effectiveness, they suffer from: 1) the explicit reasoning steps cause substantial token overhead, leading to increased computational cost; 2) the reasoning trajectory becomes fixed once the template is generated, preventing step-by-step refinement during the generation process.

Recent advances have partially addressed them by converting explicit reasoning steps into latent embeddings, enabling latent reasoning in models, such as Coconut (Hao et al., 2024). They represent the reasoning state using the LLM's hidden state (i.e., "continuous thought") and recursively feed it back into the model in the latent space to enable more effective reasoning. However, there are two critical challenges: 1) the reasoning trajectory in the latent space lacks explicit directional guidance, making it difficult to ensure consistent progression toward more accurate reasoning states; 2) the recursive embedding updates tend to be unstable, especially across multiple reasoning steps, which may compromise both robustness and accuracy. These challenges motivate us to explore how reasoning embeddings can be effectively and efficiently updated during post-training to guide the model toward more accurate solutions.

To this end, we draw inspiration from two complementary lines of research. For the challenge of providing directional guidance in reasoning embedding updates, we are inspired by reinforcement learning from human feedback (RLHF) (Ouyang et al., 2022), where learning from relative performance comparisons has demonstrated superior efficiency and effectiveness compared to relying solely on absolute supervision. For the challenge of stabilizing recursive updates, we take inspiration from the success of momentum-based optimization techniques in deep learning (Qian, 1999), which demonstrate the importance of adaptively integrating historical and current information to achieve smoother and more stable convergence.

Thus, we propose a lightweight post-training framework to refine the latent reasoning embeddings, built upon two novel strategies:

- **Contrastive Reasoning Feedback Search.** To infer updated directions in the latent reasoning space, we pass the current reasoning embedding through both a strong and a weak LLM to obtain enhanced embeddings. We derive a contrastive direction by comparing the outputs of strong and weak models, and use its gradient with respect to the current embedding to guide the reasoning embedding update.
- **Residual Embedding Refinement.** To ensure stable updates in the latent reasoning space, we blend the current reasoning embedding with its previous state using a residual weighting parameter. This interpolation smooths the transition between steps and prevents abrupt shifts in the reasoning process. As a result, the model achieves more consistent convergence across multi-step latent reasoning.

We empirically evaluate our method through comprehensive experiments and case studies, highlighting its effectiveness, efficiency, and scalability across diverse settings. These experiments demonstrate that our strategies significantly enhance reasoning performance compared to latent-only and explicit token-based reasoning baselines. Notably, on the MathQA task, our approach improves accuracy by over 5% compared to the original latent reasoning method. We further conduct case studies to illustrate how the latent embedding evolves step by step. The results show that the embedding progresses toward more accurate reasoning solutions.

These empirical findings not only validate the effectiveness of our approach but also highlight its practical value. Our framework offers three key advantages: **Efficiency and Cost-Effectiveness.** The proposed method enhances reasoning performance via a lightweight post-training refinement process. It does not require any modification to the model architecture or parameters, enabling consistent improvements with minimal cost. **Dynamic Post-Training Adaptation.** Both components operate after training to refine the reasoning process. By preserving informative latent states and exploring better latent representations, the model dynamically adjusts its internal reasoning trajectories without requiring additional training. **Training-Free Deployment.** Our refinement procedure is entirely training-free: it relies solely on forward computation in the latent space and avoids any backpropagation or parameter updates. This makes the method easy to integrate into existing models as a plug-and-play component at the post-training stage.

2 Preliminary

2.1 Problem Definition

We focus on complex reasoning tasks where a large language model (LLM) generates a correct answer y from an input question x, such as in math word problems, multi-hop question answering, and commonsense reasoning. These tasks typically require multiple inference steps, even if such steps are not explicitly annotated. Formally, the objective is to learn a function $f : x \to y$, where intermediate cognitive states are latent and only the final answer is observed. While optional intermediate steps can be included during training, they are often unavailable during inference. Building on the latent reasoning framework, we represent each reasoning step as an embedding in a latent space. Our key contribution is to model how to efficiently explore transitions within the reasoning embedding space that lead to accurate final answers. By capturing these latent trajectories, our approach enables the model to reason more effectively, even without explicit supervision over intermediate steps.

2.2 Chain-of-Thought Reasoning and Its Limitations

Chain-of-Thought (CoT) prompting (Wei et al., 2022b) and its variants (Wang et al., 2022; Yao et al., 2023b) decompose reasoning into a sequence of intermediate text steps, improving model accuracy and interpretability on complex tasks. However, CoT remains inherently limited:

- **Token-level serialization**: All reasoning steps are expressed via natural language, leading to low-dimensional, rigid representations.
- **Static trajectory**: Once the prompt is fixed, the reasoning path is deterministic, with limited room for correction.
- **Lack of feedback**: CoT does not support internal error detection or trajectory revision unless multiple sampled paths are compared externally.

These limitations motivate our shift toward latent-space reasoning. Rather than generating explicit token sequences at each step, we allow the model to evolve its internal reasoning embedding over multiple steps. This latent evolution enables the model to retain richer intermediate information, search for better latent embeddings, and adapt its reasoning process more flexibly, particularly under limited model capacity.

3 Method

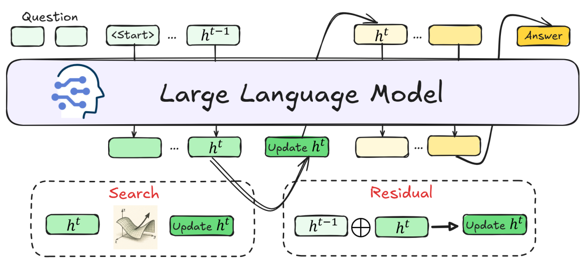


Figure 1: Overview of our reasoning framework. The bottom path shows how contrastive feedback first identifies the update direction for the reasoning embedding, which is then integrated through residual refinement to produce the current reasoning state.

Figure 1 shows the overview of our framework. We introduce a general post-training latent refinement framework that enhances latent reasoning models such as Coconut (Hao et al., 2024). Our goal is to improve reasoning stability and accuracy by augmenting the latent reasoning process with lightweight, training-free components.

In Coconut, reasoning is performed entirely in latent space without generating token-level intermediate steps, enabling compact and efficient inference. However, the model conducts latent updates in a fixed, feedforward manner, lacking the ability to revise its reasoning path or retain contextual memory across steps—limiting its adaptability when errors occur.

To address these problems, we propose a two-part refinement strategy applied after training:

- **Contrastive Reasoning Feedback Search:** This module compares reasoning outputs from a weak and a strong model to identify a contrastive improvement direction in latent space. This direction shows how the current reasoning should evolve toward stronger inference.
- **Residual Embedding Refinement:** This module integrates the contrastive feedback into the current reasoning using gated residual updates. By fusing prior context with the new signal, it preserves useful information and mitigates semantic drift across steps.

Both modules operate entirely in latent space through forward passes only, requiring no gradient updates or parameter changes. Applied during inference, they enable accurate and consistent reasoning with minimal computational overhead.

3.1 Latent Reasoning Backbone

We consider a latent reasoning framework in which the model conducts multi-step inference entirely in latent space, without producing token-level intermediate steps. As shown in the top path of **Figure 1**, the input question x is first encoded into a latent vector h^0, which is then iteratively updated for T steps using a fixed model block f:

$$h^t = f(h^{t-1}), \tag{1}$$

where $h^t \in \mathbb{R}^d$ is the latent embedding at step t. After T steps, the final state h^T is passed to a decoding head to generate the final answer. This latent-only formulation reduces token overhead and improves inference efficiency, making it particularly well-suited for small or resource-constrained models. This backbone structure forms the basis of our reasoning framework. In our implementation, we adopt Coconut (Hao et al., 2024) as the underlying latent reasoning model, where the decoder block f is derived from a pre-trained language model and kept fixed during inference.

While this setup enables compact and efficient reasoning, it still faces two key challenges:

1. **Trajectory stability**: Without memory-preserving connections, the latent trajectory may drift or collapse over time.
2. **Error correction**: There is no mechanism to guide the model back when reasoning diverges, especially in the high-dimensional latent space with many possible paths.

To address these challenges, we introduce two lightweight post-training refinement modules, contrastive latent feedback and residual embedding refinement, that operate entirely in latent space and enhance reasoning stability and correctness without any additional training.

3.2 Post-Training Latent Reasoning Refinement

The latent reasoning process described in Coconut produces a series of hidden states $h^1, h^2, ..., h^T$ by iteratively applying the model function f to an initial latent h^0. This procedure is efficient and does not generate text during reasoning. However, it performs fixed forward updates at each step and cannot revise or stabilize the reasoning trajectory.

We introduce two training-free modules that operate at the post-training stage: residual refinement and contrastive latent search. As shown in **Figure 1**, these components are applied during inference and operate on each latent state. Both modules build on the latent-only structure of Coconut and require no access to intermediate tokens. They apply directly to the latent embeddings produced in each step.

Each step starts from the output of the Coconut-style update $h^t = f(h^{t-1})$, and applies a residual preservation update followed by a contrastive adjustment. These steps are designed to stabilize and correct the latent trajectory based on human-inspired memory and comparison mechanisms.

We summarize the full inference procedure in **Algorithm 1** in the **Appendix A**.

3.2.1 Contrastive Reasoning Feedback Search

The reasoning process can still go off track even with stable latent updates due to initial uncertainty or limited model capacity. To make the system more robust, we introduce a contrastive search mechanism that enables the model to correct its latent state without any parameter updates.

As shown in **Figure 2**, we compare outputs from two models of different quality at each reasoning step t: A weaker model ("bad" model like earlier checkpoint), producing latent output h^t_{bad}; A stronger model ("good" model like later checkpoint), producing output h^t_{good}.

These outputs are generated from the same input latent h^t. The current latent embedding h^t is then updated in a direction that reduces the distance to the good model and increases the distance from the bad model. This gives the gradient signal. Then the updated latent embedding is obtained by adjusting along the contrastive direction:

$$h^t_{\text{updated}} = h^t + \eta \cdot \nabla_{h^t} \big[\text{MSE}(h^t, h^t_{\text{good}}) - \text{MSE}(h^t, h^t_{\text{bad}})\big]. \quad (2)$$

Here, $\text{MSE}(\cdot)$ denotes mean squared error between embeddings, and η is a fixed step size. This update is done through forward passes and gradient computation at the embedding level only. No model parameters are changed. The adjustment is lightweight and compatible with training-free inference, and can be applied once or iteratively depending on the step length.

This contrastive search provides a self-correction mechanism during reasoning. It helps the model adjust its latent trajectory without relying on external feedback or additional samples. This mimics the role of conflict monitoring and adjustment of the ACC in human reasoning (Botvinick et al., 2001).

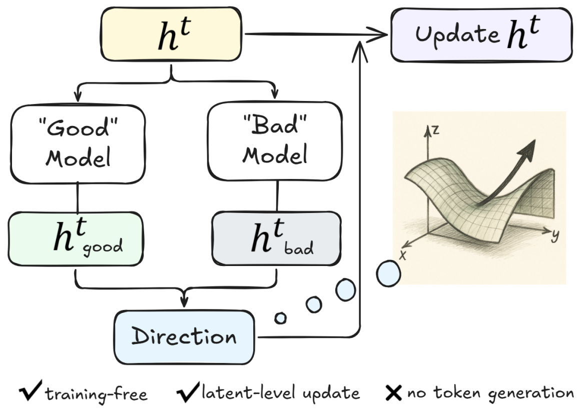


Figure 2: Contrastive Reasoning Feedback Search. We compare two models with various reasoning abilities: one stronger ("good") and one weaker ("bad"). The direction from bad to good indicates the path to move.

Coconut does not have any correction mechanism. If the reasoning goes off track, it cannot adjust or recover. CoT (Wei et al., 2022b) and Tree-of-Thoughts (Yao et al., 2023a) use external sampling or search to fix errors, but they rely on generating intermediate text. In contrast, our method updates the latent state directly using internal feedback from stronger models. This allows for efficient and flexible reasoning without relying on token-level outputs or any training.

3.2.2 Residual Embedding Refinement

At each step, we update the latent state by preserving useful information from the previous step. Instead of directly replacing the latent with the new output $f(h^{t-1})$, we blend it with the previous state h^{t-1} using a fixed-weight residual connection:

$$h^t = \alpha \cdot h^{t-1} + (1-\alpha) \cdot f(h^{t-1}), \quad \alpha \in [0,1] \quad (3)$$

where α is the memory rate. It controls how much of the previous state is kept. We use a fixed value and do not train this parameter due to the train-free setting. This design is inspired by residual networks (He et al., 2016) and resembles the working memory mechanism of the human brain (Koechlin et al., 2003). It allows the model to accumulate reasoning context over steps and prevents semantic drift. Without this refinement, the latent state may lose important early signals, which leads to an unstable reasoning trajectory.

Compared to Coconut, which discards all prior hidden states at each step, our refinement preserves and integrates context, leading to more consistent and accurate reasoning.

4 Experimental Results

We conduct experiments to evaluate whether our latent reasoning framework improves reasoning accuracy, supports training-free deployment, and operates efficiently across diverse tasks and models. Specifically, our experiments are structured to answer the following key questions: **Q1:** Can our method improve reasoning performance with minimal cost, using only training-free post-processing? **Q2:** Compared to full model retraining, can our post-training refinement achieve similar improvements with lower resource usage? **Q3:** How important are the two components—residual refinement and contrastive latent search—in contributing to performance improvement? **Q4:** Can the latent update mechanism consistently steer the model toward more accurate predictions in specific reasoning instances? **Q5:** How robust is our method to variations in key hyperparameters such as memory update rate and latent search step size? **Q6:** Can our framework generalize well across diverse language model architectures and parameter sizes?

4.1 Experimental Setup

Datasets. We evaluate our method on five representative benchmarks covering math, commonsense, and multi-hop reasoning: GSM8K (Cobbe et al., 2021), MathQA (Amini et al., 2019), AQUA-RAT (Ling et al., 2017), StrategyQA (Geva et al., 2021), and ProsQA (Hao et al., 2024). Dataset descriptions are provided in **Appendix B**.

Models. We exploit three well-recolonized open-source language models: GPT-2 (117M), Qwen-2.5 1.5B, and LLaMA-3.2 3B. For contrastive search, we utilize checkpoints from different training stages as "good" or "bad" references.

Baselines. We compare our method with the following approaches: (1) **No-CoT**: directly trains GPT-2 (Radford et al., 2019) to generate the final answer without any intermediate reasoning steps; (2)**Chain-of-Thought (CoT)** (Wei et al., 2022b): standard step-by-step natural language reasoning approach; (3) **Coconut** (Hao et al., 2024): latent reasoning without search or refinement; (4) **Ours**: latent reasoning with contrastive search and residual refinement.

Evaluation. We report exact-match accuracy, averaged over 3 random seeds. No fine-tuning is used; our method improves reasoning purely through forward latent-space updates. Full implementation details are provided in **Appendix D**.

4.2 Q1: Overall Reasoning Performance

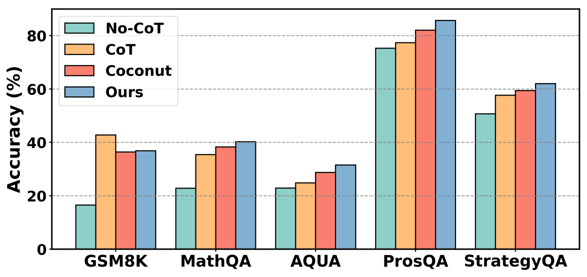


Figure 3: Accuracy (%) of different reasoning methods across five benchmarks.

To answer Q1, we compare our method with the baseline algorithms on five benchmarks to test whether our latent-space refinement strategy improves accuracy with small costs (train-free). **Figure 3** shows the reasoning accuracy (Y-axis) of our method on five benchmark tasks (X-axis).

Figure 3 shows that our method consistently outperforms latent-only reasoning (Coconut) and explicit token-based reasoning (CoT) on four out of five tasks. Notably, on the tasks of MathQA and AQUA—both that involves multi-step numerical and symbolic reasoning, our model achieves +1.95% and +2.76% absolute gains (not relative improvement ratio) over Coconut in terms of reasoning accuracy, respectively. On ProsQA and StrategyQA, which focus more on structured logical reasoning and commonsense composition, we observe absolute gains of +3.67% and +2.63% in reasoning accuracy.

The results highlight that: (1) Although CoT produces explicit thought steps, it suffers from verbosity and error accumulation, especially in non-math tasks. (2) Coconut improves reasoning by operating in a compact latent space, but lacks error correction and stable refinement. (3) Our method combines both advantages through residual preservation and feedback-driven search, thus, result into more reliable and generalizable reasoning, especially in settings where intermediate steps are implicit or hard to verbalize.

One exception is GSM8K, where CoT remains the most effective method (42.76%). This is because GSM8K contains complex arithmetic problems that require symbolic calculations. Humans rely on written steps for such tasks rather than purely mental computation. Without external tools or explicit formulas, latent reasoning struggles to handle long-chain numerical operations. In contrast, MathQA, although also math-focused, has more structured and templated problems and is multiple-choice, which makes the task easier compared to GSM8K's open-ended answers. Under such settings, our method can benefit more from embedding refinement and soft memory tracking. This highlights the differences in cognitive demands between math datasets and suggests that combining latent reasoning with symbolic or tool-augmented components may be a future direction.

4.3 Q2: Training vs. Inference Performance

To answer Q2, we compare our train-free method with other training strategies to examine reasoning accuracy and resource usages under the ProsQA dataset. We train a base model using Coconut for 30 epochs, then evaluate four settings: (1) using the original Coconut model, (2) applying our latent reasoning only during inference, (3) continuing training for 10 more epochs with our latent reasoning enabled, and (4) applying our latent reasoning during both training and inference.

Figure 4 shows that our method achieves the best accuracy (+4.47%) when applied only during inference, with minimal overhead (24 seconds, 31.23GB memory). In contrast, continuing training with latent reasoning adds significant time (54+ minutes) and memory cost (39.04GB), yet leads to smaller accuracy gains (+1.63%). Using the method in both training and inference further increases the cost but gives almost no improvement. These results highlight the efficiency of our strategy: without any backpropagation or weight updates, our inference-only setup improves performance while saving training time and GPU resources and without the need of further training.

Insights. Our method avoids expensive retraining, requires only forward computation, and still brings accuracy improvement. It is lightweight and improves the reasoning trajectory without changing the model parameters. These computational benefits make it practical for deployments in low-resource or frozen-model scenarios.

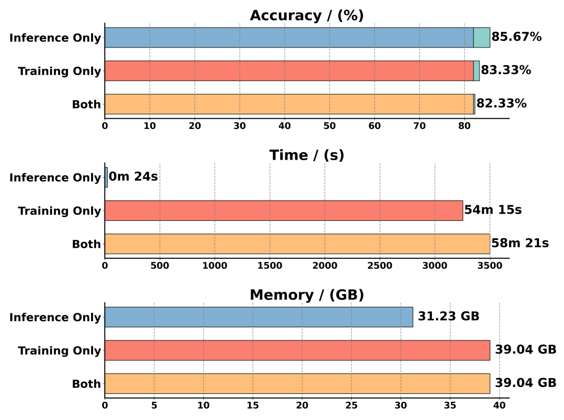


Figure 4: Latent Reasoning in Training vs. Inference.

4.4 Q3: Study of Contrastive Search and Residual Refinement

To answer Q3, we remove contrastive search and residual refinement one by one to understand the role of each technical component.

Table 1: Ablation study on MathQA. We show the impact of each component in our method.

Variant	Accuracy (%)	Gain (%)
Latent only	38.25	-
+ Residual refinement	40.02	+4.63
+ Latent Search	39.79	+4.03
+ Residual + Search (ours)	**40.20**	**+5.10**

Table 1 highlights the importance of both residual refinement and contrastive search in our framework. Compared with the baseline Coconut with only latent thoughts, our method of incorporating residual connections yields an +4.63% improvement in accuracy. This observation demonstrates that preserving and gradually refining previous latent states help to stabilize reasoning trajectories. It aligns with the working memory mechanism in human prefrontal cortex, where ongoing cognitive representations are maintained and adjusted over time. Only incorporating contrastive search leads to a gain of +4.03%. This observation shows that self-correction based on improving directions, which mimic the role of the anterior cingulate cortex (ACC) in conflict detection, enables the model to recover from suboptimal reasoning directions. Finally, integrating both residential connections and contrastive search can result in the best performance (+5.10%). This observation shows that stable memory evolution and dynamic search together form a lightweight and training-free latent reasoning mechanism that maintains contexts, detects errors, and refines internal representations without relying on explicit intermediate language.

4.5 Q4: A Step-wise Case Study

To answer Q4, we visualize the impacts of latent reasoning refinement on answer prediction to better understand how our method improves reasoning using the MathQA dataset.

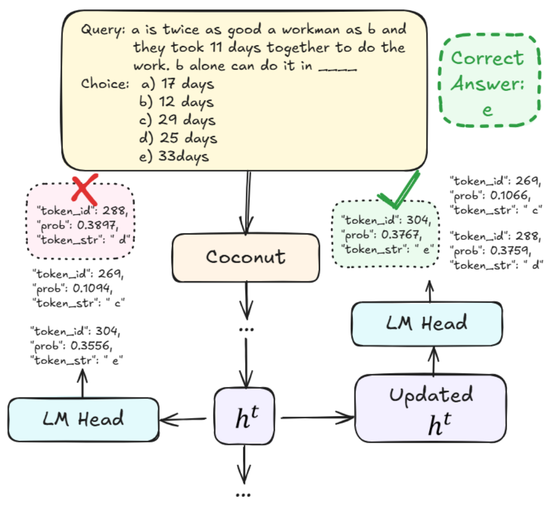
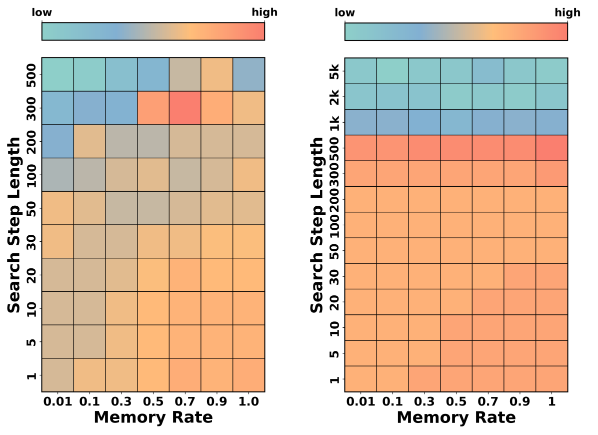

Query: a is twice as good a workman as b and they took 11 days together to do the work. b alone can do it in ____
Choice: a) 17 days
b) 12 days
c) 29 days
d) 25 days
e) 33days
Correct Answer: e
"token_id": 288, "prob": 0.3897, "token_str": " d"
"token_id": 269, "prob": 0.1094, "token_str": " c"
"token_id": 304, "prob": 0.3556, "token_str": " e"
Coconut
"token_id": 304, "prob": 0.3767, "token_str": " e"
"token_id": 269, "prob": 0.1066, "token_str": " c"
"token_id": 288, "prob": 0.3759, "token_str": " d"
LM Head
h^t
Updated h^t
LM Head

Figure 5: Case study: our method adjusts the latent embedding to reach the correct answer.



(a) Sensitivity on MathQA (b) Sensitivity on AQUA

Figure 6: Hyperparameter sensitivity on two datasets. The colors measure reasoning accuracy.

Figure 5 shows the baseline Coconut produces a latent state h_t and predicts the wrong choice ("d") with the highest probability. After applying our latent refinement, the updated embedding h'_t leads to the correct prediction ("e"). One possible explanation is: contrastive search adjusts the latent state using information from reference models, while residual refinement helps to preserve internal information across steps. Both are used in forward steps only without additional training. The model dynamically adjusts its internal representation to align with the correct reasoning trajectory.

Insights. The case study illustrates how our method enables internal latent correction before decoding. Rather than relying on external tokens or explicit logic, the model self-adjusts in latent space, which reflects how humans reconsider their thoughts before answering. We provide three more examples in **Appendix G**.

4.6 Q5: Study of Hyperparameter Sensitivity

To answer Q5, we examine the sensitivity of two key parameters of contrastive search and residual refinement using the MathQA and AQUA datasets: (1) **Search Step Length** (η in **Equation** (2)): the step of each latent update in the contrastive search; (2) **Memory Rate** (α in **Equation** (3)): how much previous latent state is memorized across reasoning steps. For each query, we perform 3 rounds of latent refinement, each of which includes one residual update and one search update, and fix other factors for fair comparisons.

Figure 6a shows the heatmap of accuracy on MathQA over memory rates and search steps, where red indicates high accuracy and green indicates low accuracy. Overall, the reasoning accuracy increases when the memory rate increases, but is less affected by the search step length. This suggests that mathematical problem solving benefits more from stable accumulation of prior steps and residual refinement (i.e., memory rate). A stronger memory allows the model to preserve intermediate reasoning steps for better reasoning. **Figure 6b** shows the opposite trend. AQUA is less sensitive overall but is more affected by the search step size. This suggests that commonsense QA relies more on adaptive correction than long-term memory, so flexible updates in latent space have a larger impact than memory accumulation.

Insights. The results show that different reasoning tasks may rely on different cognitive mechanisms. Math-heavy tasks need better memory. QA tasks benefit more from flexible error correction. Our framework allows both without extra training.

4.7 Q6: Generalization Across Backbones

To answer Q6, we evaluate the generalization of our framework on different LLM backbones across architectures and sizes: GPT-2 (117M), Qwen-2.5 1.5B, and LLaMA-3.2 1B models. **Table 2** shows that our method on top of the three backbone models reach over 40% accuracy on MathQA. This suggests that our latent refinement strategy can be applied to both small and medium-sized models without any changes to model architecture. Our method adds little overhead in terms of resource usage. Meanwhile, training time and memory usage increase with model size as expected, but inference remains efficient. All models finish complete in under 7 minutes with less than 24 GB of memory. Our method is simple and general. It can be used on top of any existing language model to improve reasoning performance, even when training resources are limited. It does not require retraining or modification of model weights, which makes it easy to adopt in different settings.

Table 2: Comparison across LLMs on MathQA.

	GPT-2	Qwen	LLaMA
Parameters	117M	1.5B	3B
Accuracy (%)	40.20	43.04	41.10
Train Time (min)	18.95	95.80	78.38
Train Mem (GB)	45.68	45.33 (LoRA)	45.74 (LoRA)
Infer Time (min)	4.27	6.31	5.83
Infer Mem (GB)	22.64	23.12	22.84

4.8 Study of Token Efficiency

Latent reasoning avoids generating intermediate natural language steps (i.e., thoughts), thus, reduce output token number. We compare the average number of generated tokens between Chain-of-Thought (CoT) prompting and our latent reasoning method on the MathQA and AQUA datasets. **Table 3** shows that latent method reduces token usage by over 92% on both datasets. This is because latent steps are performed in a latent space and don't produce textual outputs at each reasoning step. These results suggest that latent reasoning is not only more compact but also more cost-efficient during inference. This makes it particularly suitable for deployment in resource-constrained settings or large-scale usage.

Table 3: Average generated tokens per query.

	CoT	Latent	Reduction
MathQA	66.71	5.02	92.47%
AQUA	72.73	5.31	92.65%

5 Related Work

5.1 Chain of Thought Reasoning

Chain-of-Thought (CoT) prompting enhances reasoning in LLMs by decomposing complex problems into step-by-step textual reasoning traces (Wei et al., 2022b; Kojima et al., 2022). Later works improve CoT with better aggregation (e.g., self-consistency (Wang et al., 2022)), scalable prompt generation (Zhang et al., 2022), and tree-structured exploration like Tree-of-Thoughts (Yao et al., 2023a). Prompting strategies have also been optimized through complexity-aware design (Fu et al., 2022) and iterative demonstration (Nye et al., 2021). These techniques have been widely applied to domains such as math (Zhou et al., 2022), commonsense (Huang et al., 2022), and symbolic logic (Liu et al., 2023), showing broad improvements. Extensions such as Selection-Inference (Creswell et al., 2022), Program-of-Thoughts (Chen et al., 2022), and Multimodal-CoT (Lu et al., 2022) further combine CoT with selection mechanisms, symbolic programs, or visual reasoning inputs.

5.2 Latent-Space Reasoning in LLMs

Recent work proposes moving reasoning from token-level generation to latent space updates, enabling more compact and abstract reasoning. Coconut (Hao et al., 2024) is a foundational method that replaces intermediate token outputs with internal latent states that evolve step by step. Other approaches expand this idea through latent sampling (LaTRO (Chen et al., 2024)), self-training to uncover latent reasoning (SERT (Zhang et al., 2025)), or expectation-maximization loops for iterative reasoning (Ruan et al., 2025). Looped transformers (Saunshi et al., 2025) reuse a smaller model multiple times to simulate deep reasoning trajectories. In applied domains, ReaRec (Tang et al., 2025) introduces latent multi-step reasoning into recommender systems, demonstrating the broader utility of latent-state methods beyond language tasks.

6 Conclusion Remarks

We present a latent reasoning framework for LLM by introducing residual refinement and contrastive latent search to improve stability and enable dynamic self-correction without retraining. The new framework operates in a training-free, plug-and-play manner. Experiments show that our method improves accuracy by 2–5% over latent-only reasoning across five benchmarks, with up to 7.7% gain on ProsQA. The improvement demonstrates the effectiveness of latent refinement with internal information. We show that even without token-level outputs or supervised feedback, LLMs can adjust internal belief states using post-hoc representational updates, pointing toward a new direction of self-corrective, embedding-level reasoning.

Limitations

While our framework improves reasoning performance across multiple tasks and models, it still has several limitations. (1) Even though the method can be applied to almost all open-source LLMs, we just test it on some relatively small LLMs. So the results are not comparable to the latest model, like ChatGPT-4.1 or DeepSeek-R1. In addition, we do not compare with sampling-based or search-based CoT variants (e.g., Tree-of-Thought (Yao et al., 2023a), Self-Consistency (Wang et al., 2022)), since our focus is on deterministic, internal latent refinement without additional decoding or reranking. (2) Latent reasoning may not be optimal for all types of tasks. For example, human tend to solve math problems with written-down logic, which involves explicit reasoning processes. In the future study, the routing method (Wang et al., 2025) can be applied to dynamically reason the query. (3) Latent reasoning lacks human-readable intermediate steps, making it harder to interpret or debug compared to token-based methods. (4) Inspired by LLM agent researches, the future work can introduces tools to ensure stronger, more correct, and more reasonable reasoning processes.

References

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Matthew M Botvinick, Todd S Braver, Deanna M Barch, Cameron S Carter, and Jonathan D Cohen. 2001. Conflict monitoring and cognitive control. *Psychological review*, 108(3):624.

Haolin Chen, Yihao Feng, Zuxin Liu, Weiran Yao, Akshara Prabhakar, Shelby Heinecke, Ricky Ho, Phil Mui, Silvio Savarese, Caiming Xiong, and 1 others. 2024. Language models are hidden reasoners: Unlocking latent reasoning capabilities via self-rewarding. *arXiv preprint arXiv:2411.04282*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

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*.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*.

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.

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*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pages 9118–9147. PMLR.

Etienne Koechlin, Chrystele Ody, and Frédérique Kouneiher. 2003. The architecture of cognitive control in the human prefrontal cortex. *Science*, 302(5648):1181–1185.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Hanmeng Liu, Zhiyang Teng, Ruoxi Ning, Jian Liu, Qiji Zhou, and Yue Zhang. 2023. Glore: Evaluating logical reasoning of large language models. *CoRR*.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, and 1 others. 2021. Show your work: Scratchpads for intermediate computation with language models.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

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.

Yangjun Ruan, Neil Band, Chris J Maddison, and Tatsunori Hashimoto. 2025. Reasoning to learn from latent thoughts. *arXiv preprint arXiv:2503.18866*.

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*.

Jiakai Tang, Sunhao Dai, Teng Shi, Jun Xu, Xu Chen, Wen Chen, Wu Jian, and Yuning Jiang. 2025. Think before recommend: Unleashing the latent reasoning power for sequential recommendation. *arXiv preprint arXiv:2503.22675*.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. 2025. MixLLM: Dynamic routing in mixed large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10912–10922, Albuquerque, New Mexico. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, and 1 others. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Yong Zhang, Bingyuan Zhang, Zhitao Li, Ming Li, Ning Cheng, Minchuan Chen, Tao Wei, Jun Ma, Shaojun Wang, and Jing Xiao. 2025. Self-enhanced reasoning training: Activating latent reasoning in small models for enhanced reasoning distillation. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and 1 others. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Algorithm Overview

Here is the Algorithm pseudo-code of our train-free framework.

Algorithm 1 Latent Reasoning with Residual Refinement and Contrastive Search (Inference-Time)

```
Require: Question input $x$, reasoning steps $T$, pretrained model $f$, residual weight $\alpha$, search step size $\eta$, strong model $f_{\text{good}}$, weak model $f_{\text{bad}}$
 1: Encode input $x$ into initial latent state $h^0$
 2: for $t = 1$ to $T$ do
 3:     Compute latent update: $\Delta h^t \leftarrow f(h^{t-1})$
 4:     Residual refinement: $h^t \leftarrow \alpha \cdot h^{t-1} + (1 - \alpha) \cdot \Delta h^t$
 5:     if Contrastive search is enabled then
 6:         Get strong model output: $h^t_{\text{good}} \leftarrow f_{\text{good}}(h^t)$
 7:         Get weak model output: $h^t_{\text{bad}} \leftarrow f_{\text{bad}}(h^t)$
 8:         Compute gradient: $g^t \leftarrow \nabla_{h^t} \left[\text{MSE}(h^t, h^t_{\text{good}}) - \text{MSE}(h^t, h^t_{\text{bad}})\right]$
 9:         Contrastive update: $h^t \leftarrow h^t + \eta \cdot g^t$
10:     end if
11: end for
12: Decode final latent $h^T$ into answer $y$
13: return Answer $y$
```

B Dataset Descriptions

GSM8K (Cobbe et al., 2021)[1]: A dataset of 8.5K high-quality grade school math word problems created by human problem writers. It is divided into 7.5K training and 1K test examples. Each problem requires 2–8 steps of basic arithmetic reasoning (addition, subtraction, multiplication, division) and includes a detailed natural language solution explanation.

MathQA (Amini et al., 2019)[2]: A dataset containing approximately 37.2K math word problems derived from AQuA. Each problem is annotated with a programmatic solution template using a domain-specific language comprising 58 operators. The dataset covers various mathematical domains, including percentages, geometry, and linear equations, and provides multiple-choice answers.

AQUA-RAT (Ling et al., 2017)[3]: A large-scale dataset consisting of approximately 100K algebraic word problems. Each question is accompanied by a step-by-step natural language rationale explaining the solution process. The dataset is designed to train models capable of generating both the solution and the explanatory rationale.

StrategyQA (Geva et al., 2021)[4]: A question-answering benchmark comprising 2,780 examples. Each question requires implicit multi-hop reasoning, where the necessary reasoning steps are not explicitly stated and must be inferred. The dataset includes decompositions and evidence paragraphs for each question.

ProsQA (Hao et al., 2024): A diagnostic dataset designed to evaluate models' planning abilities. It features examples that require multi-step reasoning and the ability to handle distractors. Each question is paired with a human-written rationale, facilitating the assessment of models' reasoning processes.

C Chain-of-Thought Data Construction

To compare different reasoning paradigms, we construct CoT-style training and evaluation data from datasets that provide intermediate reasoning steps. Each example contains three parts: a **question** x, a list of **steps** $s = \{s_1, s_2, ..., s_n\}$ describing the reasoning trace, and a final **answer** y.

[1]https://huggingface.co/datasets/openai/gsm8k

[2]https://huggingface.co/datasets/allenai/math_qa

[3]https://huggingface.co/datasets/deepmind/aqua_rat

[4]https://huggingface.co/datasets/voidful/StrategyQA

We format the data differently depending on the method:

- **No-CoT (Direct QA)**: The model is trained to map the input question directly to the answer. Only (x, y) pairs are used, without any reasoning trace.
- **CoT (Textual Supervision)**: The model is trained to generate both the reasoning steps and the final answer as a single sequence. The input is x, and the output is $\texttt{join}(s_1, ..., s_n) + y$. This helps the model learn to reason explicitly in text.
- **Latent CoT (Latent Reasoning)**: We follow the Coconut (Hao et al., 2024) setup. The model is trained to generate the answer y from input x, but the reasoning steps s are used as supervision in the latent space. They are not generated as tokens, but guide the intermediate hidden representations.

This unified setup allows us to study how different forms of reasoning supervision affect model performance, and how latent-space reasoning compares to explicit token-level chains.

D Implementation and Hyperparameters

Optimization. All models are trained with the AdamW optimizer, using a learning rate of 3×10^{-5}, batch size 16, and weight decay 0.01. Training runs for 50 epochs unless specified otherwise. When training is staged (e.g., for latent reasoning), we use 30 epochs for initial training and 10 for optional refinement. Gradients are accumulated every step (no gradient accumulation).

Model Backbones. We primarily use GPT-2 (117M) as the language model. The pretrained model is loaded from `openai-community/gpt2`. For contrastive latent search, we define a "bad" checkpoint (early-stage model) and a "good" checkpoint (later-stage model), loaded separately and used in forward-only search (no weight update).

Latent Reasoning Configuration. All reasoning happens in the latent space:

- **Latent update steps:** 3 iterations per query.
- **Search step size** (η)**:** from 1 to 5000: controls the gradient step in contrastive search.
- **Residual memory rate** (α)**:** from 0.01 to 1: controls how much information from the previous latent state is retained.

Both residual update and contrastive search are applied at each reasoning step without gradient backpropagation. The full process is training-free and runs with forward computation only.

Tokenization and Decoding. We use the native tokenizer of each backbone model. For latent reasoning, we add special tokens such as `<|latent|>` to mark latent segments. The max token limit is set to 64 for math tasks and 128 for QA tasks.

Reproducibility. All experiments are run with seed 0. Our setup supports distributed training and evaluation across 4 GPUs. Each configuration, model path, and dataset split is version-controlled and specified in the released code.

E Experimental Environment

All experiments were conducted on the Ubuntu 22.04.3 LTS operating system, with a 13th Gen Intel(R) Core(TM) i9-13900KF CPU, 128GB of system RAM, and four NVIDIA RTX A6000 GPUs (each with 48GB of VRAM). The experiments were implemented using Python 3.11.5 and PyTorch 2.0.1.

F Generalization to Unseen Tasks

In this section, we analyze whether latent reasoning learned from one task can generalize to another without re-training. This evaluates the portability of internal latent-space dynamics across domains.

We conducted a controlled experiment where all models were trained on GSM8K and directly tested on MathQA. Both datasets involve multi-step math word problems but differ in format, vocabulary, and complexity. For reference, we also evaluate models trained directly on MathQA.

Table 4: Generalization to MathQA: Accuracy of models trained on different datasets

Model	Trained on MathQA	Trained on GSM8K	Difference
CoT	35.42	0.00	-35.42
Coconut	38.25	0.00	-38.25
Ours	**40.20**	0.00	**-40.20**

Observation. As shown in Table 4, none of the models generalize successfully from GSM8K to MathQA. Accuracy drops to 0 across CoT, Coconut, and our latent reasoning framework.

Analysis. These results highlight the challenge of cross-task generalization in small and medium-sized LLMs. Despite using latent representations and post-training refinement, our method—like CoT and Coconut—fails to transfer reasoning skills between datasets. We believe this is due to two factors: (1) models like GPT-2 lack the scale and capacity needed for emergent generalization (Wei et al., 2022a); (2) our method is designed to refine reasoning trajectories within a task, not to perform domain adaptation. While effective in-distribution, the framework does not substitute for task-specific training when reasoning patterns differ across domains.

G More Output Case Studies

We present additional examples to illustrate how our latent reasoning refinement improves model predictions. In each case, the base Coconut model produces a suboptimal answer, while our method adjusts the latent representation to arrive at the correct one. These examples further demonstrate the effectiveness of residual and contrastive updates at inference time.

G.1 Example 1

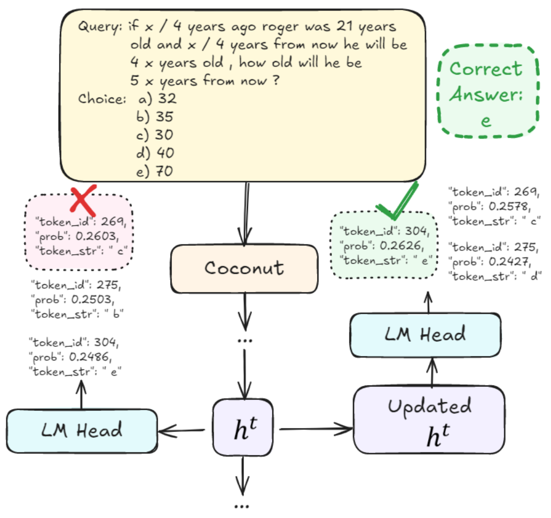

Query: if x / 4 years ago roger was 21 years old and x / 4 years from now he will be 4 x years old , how old will he be 5 x years from now ?
Choice: a) 32
b) 35
c) 30
d) 40
e) 70
Correct Answer: e
"token_id": 269, "prob": 0.2603, "token_str": " c"
"token_id": 275, "prob": 0.2503, "token_str": " b"
"token_id": 304, "prob": 0.2486, "token_str": " e"
Coconut
...
LM Head
h^t
...
"token_id": 304, "prob": 0.2626, "token_str": " e"
"token_id": 269, "prob": 0.2578, "token_str": " c"
"token_id": 275, "prob": 0.2427, "token_str": " d"
LM Head
Updated h^t

Figure 7: Additional Case study 1: our method adjusts the latent embedding to reach the correct answer.

In **Figure 7**, the input question requires multi-hop reasoning. The base model (Coconut) assigns the highest probability to the incorrect option "c" (0.2603), with the correct answer "e" ranked third (0.2486). After our latent refinement procedure, the representation is updated, and the model now predicts "e" with the highest confidence (0.2626), correcting its earlier mistake. This case shows that even small adjustments in the latent space can significantly shift the model's final decision.

G.2 Example 2

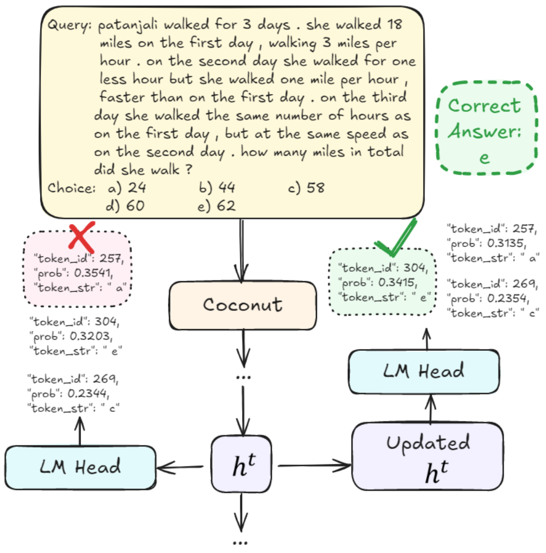

Query: patanjali walked for 3 days . she walked 18 miles on the first day , walking 3 miles per hour . on the second day she walked for one less hour but she walked one mile per hour , faster than on the first day . on the third day she walked the same number of hours as on the first day , but at the same speed as on the second day . how many miles in total did she walk ?
Choice: a) 24 b) 44 c) 58 d) 60 e) 62
Correct Answer: e
"token_id": 257, "prob": 0.3541, "token_str": " a"
"token_id": 304, "prob": 0.3203, "token_str": " e"
"token_id": 269, "prob": 0.2344, "token_str": " c"
Coconut
...
LM Head
h^t
...
"token_id": 304, "prob": 0.3415, "token_str": " e"
"token_id": 257, "prob": 0.3135, "token_str": " a"
"token_id": 269, "prob": 0.2354, "token_str": " c"
LM Head
Updated h^t

Figure 8: Additional Case study 2: our method adjusts the latent embedding to reach the correct answer.

Figure 8 contains a more complex temporal and numerical reasoning question. Initially, the model incorrectly predicts option "a" (0.3541) as the answer. After our latent-space update, the probability of "e" rises to 0.3491, making it the top prediction. The model successfully incorporates the revised latent information to override its earlier bias toward an incorrect answer. This example underscores the ability of latent search to shift the model's internal belief in a non-disruptive and interpretable way.

G.3 Example 3

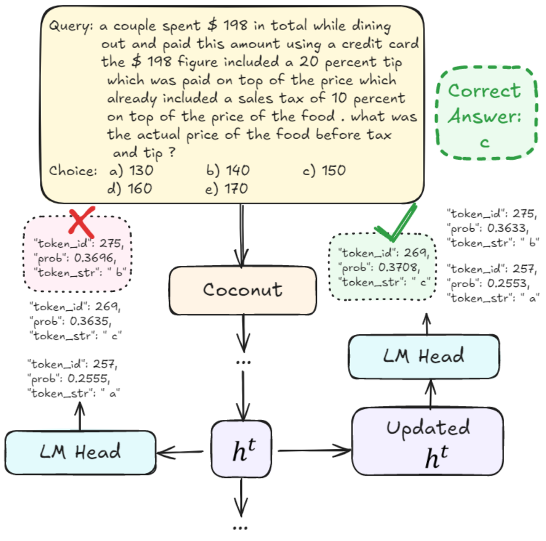

Query: a couple spent $ 198 in total while dining out and paid this amount using a credit card the $ 198 figure included a 20 percent tip which was paid on top of the price which already included a sales tax of 10 percent on top of the price of the food . what was the actual price of the food before tax and tip ?
Choice: a) 130 b) 140 c) 150 d) 160 e) 170
Correct Answer: c
"token_id": 275, "prob": 0.3696, "token_str": " b"
"token_id": 269, "prob": 0.3635, "token_str": " c"
"token_id": 257, "prob": 0.2555, "token_str": " a"
Coconut
...
LM Head
h^t
...
"token_id": 269, "prob": 0.3708, "token_str": " c"
"token_id": 275, "prob": 0.3633, "token_str": " b"
"token_id": 257, "prob": 0.2553, "token_str": " a"
LM Head
Updated h^t

Figure 9: Additional Case study 3: our method adjusts the latent embedding to reach the correct answer.

In **Figure 9**, the model is tasked with a multi-step calculation problem involving percentages and sales tax. The initial output ranks "b" highest (0.3696), but the correct answer "c" is only second (0.3635). After applying contrastive search, the updated latent embedding increases the score of "c" to 0.3708, making it the new top choice. Notably, this correction occurs without any retraining or additional supervision.