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

**Xinyuan Wang[1], Dongjie Wang[2], Wangyang Ying[1], Haoyue Bai[1], Nanxu Gong[1], Sixun Dong[1], Kunpeng Liu[3], Yanjie Fu[1]***

[1]Arizona State University, Tempe, USA
[2]University of Kansas, Lawrence, USA
[3]Clemson University, Clemson, USA
{xwang735, wying4, haoyueba, ngong6, sdong46, yanjie.fu}@asu.edu, wangdongjie@ku.edu, kunpenl@clemson.edu

## 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** — https://github.com/anord-wang/Lateng-Reasoning

## 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. 2022) 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 intermediate 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 model to obtain enhanced embeddings. We derive a contrastive direction by comparing the outputs of strong and weak models, and use its gradient concerning the current embedding to guide the embedding update. The strong model is only better relative to the weak one, and does not limit the performance of the final method.
- **Residual Embedding Refinement.** To ensure stable updates in the latent space, we blend the current reasoning embedding with its previous state using a residual weighting strategy. This interpolation smooths the transition between steps and prevents abrupt shifts in the reasoning process. As a result, the model achieves more con-

*Corresponding author.

sistent convergence across multi-step latent reasoning.

We 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.

## Preliminary

### Problem Definition

We focus on complex reasoning tasks where a large language model (LLM) aims to generate 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 $f : x \rightarrow y$, 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.

### Chain-of-Thought Reasoning and Its Limitations

Chain-of-Thought (CoT) prompting (Wei et al. 2022) 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 Chain-of-Thought 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 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.

## Method

**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. This end-to-end process and training cannot revise its reasoning path or retain contextual memory across steps during inference time, which limits its adaptability when errors occur. Adding such a correction capability during training would require large amounts of data and extensive optimization, making it expensive and less practical.

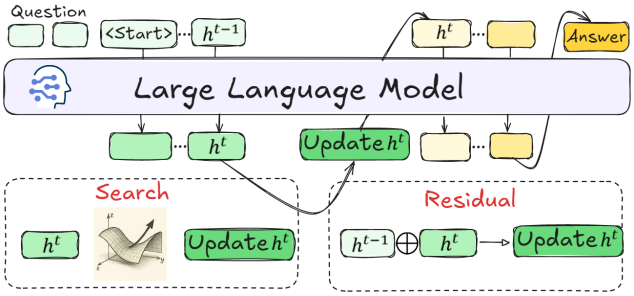



Figure 1: Overview of our reasoning framework. The bottom path shows how contrastive feedback first identifies the update direction for the reasoning embedding, which is then integrated through residual refinement to produce the current reasoning state.

To address these problems, we propose a two-part refinement strategy applied in the post-training stage:

- **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.

This strategy is lightweight and training-free. Both modules operate entirely in latent space using forward passes only, without any gradient updates or parameter changes. It allows Coconut to adjust its reasoning during inference by following a contrastive direction, while the residual update prevents overly large changes, helping to retain useful information. Applied at inference time, these components enable more accurate and stable reasoning with minimal computational overhead.

## 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 $x \in \mathbb{R}^L$ is first encoded into a latent embedding $h^0$, which is then iteratively updated for $T$ steps using a fixed model block $f$:

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

where $h^t \in \mathbb{R}^{L \times d}$ 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 pretrained language model and kept fixed during inference.

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

1. **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.
2. **Trajectory stability**: In multi-step reasoning, the latent trajectory may drift without memory-preserving connections, leading to unstable or inconsistent reasoning.

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.

## 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.

**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 the outputs from two models of different quality at each reasoning step $t$: a weaker model ("bad" model, such as an early checkpoint) producing latent output $h^t_{\text{bad}}$, and a stronger model ("good" model, such as a later checkpoint) producing output $h^t_{\text{good}}$.

These two models are not the final model used for latent reasoning inference (e.g., Coconut), but are intermediate checkpoints saved during the training of a standard CoT model. The distinction between "good" and "bad" is relative and only exists for the purpose of creating a directional signal in latent space. The goal is to identify a better direction to refine the current reasoning state, not to imitate or match the stronger model's behavior.

Importantly, the "good" model used in this comparison is not stronger than our final model. In fact, both "good" and "bad" models are weaker than Coconut. Their role is solely to help define an updated direction based on contrast, rather than to serve as a performance reference or target.

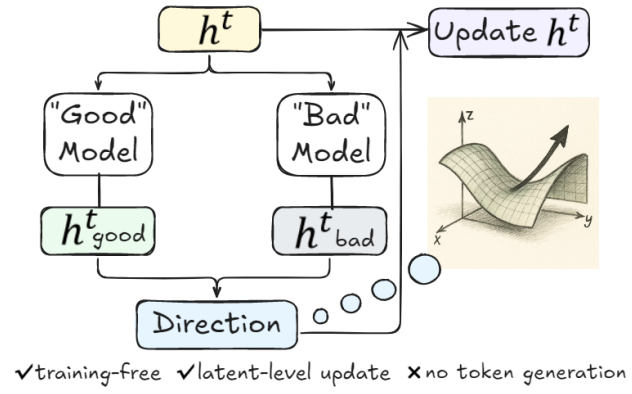



Figure 2: Contrastive Reasoning Feedback Search. We compare two models with various reasoning abilities: one stronger ("good") and one weaker ("bad"). The direction from bad to good indicates the path to move.

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 gra-

dient signal. Then the updated latent embedding is obtained by adjusting along the contrastive direction:

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

Here, $\text{MSE}(\cdot)$ denotes mean squared error between embeddings, and $\eta$ 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 ACC adjustment in human reasoning (Botvinick et al. 2001).

Currently, Coconut does not have any correction mechanism. If the reasoning goes off track, it cannot adjust or recover. CoT (Wei et al. 2022) 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 contrastive direction. This is efficient and flexible, without relying on token-level outputs or any training.

**Residual Embedding Refinement** To make the latent embeddings evolve stably, we add a residual mechanism when updating the embeddings. 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 $\alpha$ 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, Ody, and Kouneiher 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.

In training-free settings, we cannot rely on backward gradients to correct mistakes, so it becomes more important to preserve helpful signals across steps. The residual update plays this role by softly carrying forward past reasoning information, making the trajectory more stable. While the contrastive reasoning search explores better directions in latent space, the residual refinement helps maintain stability during multi-step updates.

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.

# 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 fully retraining, can our post-training refinement achieve 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 changes in hyperparameters like memory update rate and latent search step length?

**Q6:** Can our framework generalize well across diverse language model architectures and parameter sizes?

## 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).

**Models.** We use GPT-2 base (117M) to conduct the majority of experiments. We also try two other well-recognized open-source language models: Qwen-2.5 1.5B and LLaMA-3.2 3B. For contrastive search, we utilize checkpoints from different training stages in the CoT method 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. 2022): 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 precision, averaged on 3 random seeds. No fine-tuning is used; our method improves reasoning only through forward latent-space updates.

## Q1: Overall Reasoning Performance

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

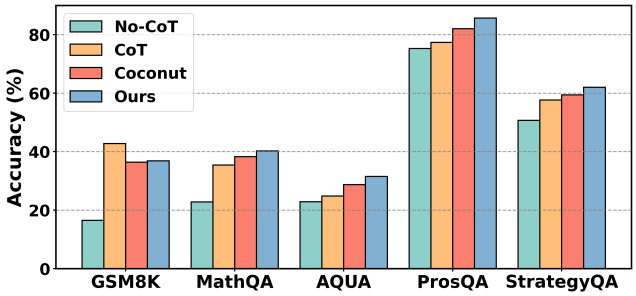



Figure 3: Accuracy (%) of different reasoning methods across five benchmarks.

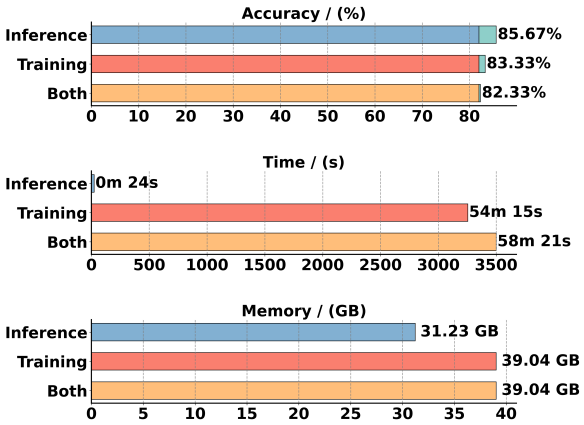



Figure 4: Latent Reasoning in Training vs. Inference.

+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 resulting 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.

## Q2: Training vs. Inference Performance

To answer Q2, we compare our train-free method with other training strategies to examine the accuracy of reasoning and the usage of resources in the ProsQA dataset. We train a base Coconut model for 30 epochs, then evaluate four settings: (1) using the original Coconut model, (2) only applying the latent reasoning during inference, (3) continuing training for 10 more epochs with our latent reasoning, 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. This result may seem counter-intuitive at first. But we observed the same pattern on three datasets, so we believe it is consistent. Our understanding is: during training, the refinement only affects the forward pass. It does not go into the backward gradient, so the model may not fully learn how to use it. With limited training data, this extra signal may even confuse the model or hurt convergence. Then, during inference, if we apply refinement again, the model receives a second adjustment. This may "overshoot" the correct direction and make the reasoning worse. That could explain why training + inference gives smaller gains than inference-only.

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 for further training.

**Insights.** Our method avoids expensive re-training, 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.

## Q3: Study of Contrastive Search and Residual Refinement

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

**Table 1** highlights the importance of residual refinement and contrastive search in our framework. Compared to the Coconut baseline with only latent thoughts, incorporating residual connections yields a +4.63% improvement in accuracy. This suggests that preserving and gradually refining previous latent states helps stabilize reasoning trajectories across steps. Using only contrastive search leads to

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

a +4.03% gain, showing that updating the current latent state with better directions enables the model to recover from suboptimal reasoning. Finally, integrating both residential connections and contrastive search results in the best performance (+5.10%). This 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 tokens.

## 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.

**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.

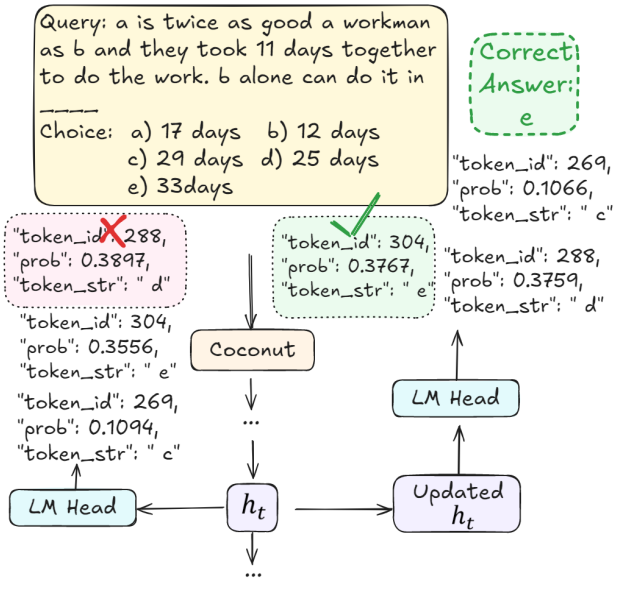



Figure 5: Case study: our method adjusts the latent embedding to reach the correct answer.

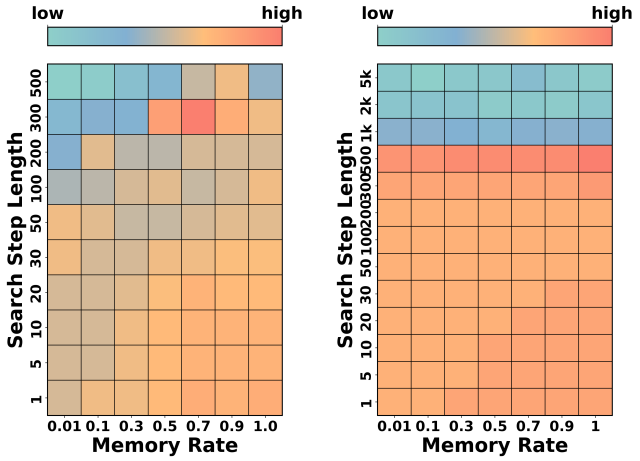
**Insights.** The case study illustrates how our method enables internal latent correction before decoding. Rather than relying on external tokens or explicit logic, the model self-adjusts in latent space, which reflects how humans reconsider their thoughts before answering.

## 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** ($\eta$ in **Equation** (2)): the step of each latent update in the contrastive search; (2) **Memory Rate** ($\alpha$ 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 heat map of accuracy in MathQA on 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 solving mathematical problems 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 common-sense QA relies more on adaptive correction than long-term memory, so flexible updates in latent space have a greater 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.




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

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

### Q6: Generalization Across Backbones

To answer Q6, we evaluate the generalization of our framework on different LLM backbones across architectures and sizes: GPT-2 base (117M), Qwen-2.5 1.5B, and LLaMA-3.2 1B models.

| | GPT-2 | Qwen | LLaMA |
|---|---|---|---|
| Parameters | 117M | 1.5B | 3B |
| Accuracy (%) | 40.20 | 43.04 | 41.10 |
| Latent Only Acc (%) | 38.25 | 42.29 | 39.30 |
| Gain (%) | +5.10 | +1.77 | +4.58 |
| 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 |

Table 2: Comparison across LLMs on MathQA.

**Table 2** shows that our method improves accuracy over the baseline on all three backbones. For example, on Qwen and LLaMA, the base accuracy is 42.29% and 39.30%, and our method improves it to 43.04% and 41.10%, with gains of +1.77% and +4.58%, respectively. This confirms that our latent refinement strategy can work across different model architectures and sizes.

The method is simple and general. It does not require any changes to the model architecture and applies equally to small and medium models. It also adds little overhead. Training time and memory increase with model size as expected, but inference remains efficient — all models complete in under 7 minutes and under 24 GB of memory.

Our approach is especially practical for frozen or resource-limited setups. It can be used on top of any existing language model to improve reasoning, without re-training or modifying model weights.

### 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.

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

Table 3: Average generated tokens per query.

**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.

## Related Work

### 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. 2022; 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, Shanahan, and Higgins 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.

### 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.

## 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.

Our method is simple, efficient, and general. It works across different model architectures and sizes, and adds minimal overhead during inference. Even without generating intermediate text or using extra supervision, LLMs can still adjust their internal states through small updates in the latent space. This shows a new way to make models correct themselves at the embedding level, which can be useful for frozen models or low-resource settings.

## Acknowledgments

Dr. Yanjie Fu is supported by the National Science Foundation (NSF) via the grant numbers: 2426340, 2416727, 2421865, 2421803. Dr. Kunpeng Liu is supported by the National Science Foundation (NSF) via the grant numbers 2550105, 2550106, and 2242812.

## References

Amini, A.; Gabriel, S.; Lin, P.; Koncel-Kedziorski, R.; Choi, Y.; and Hajishirzi, H. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Botvinick, M. M.; Braver, T. S.; Barch, D. M.; Carter, C. S.; and Cohen, J. D. 2001. Conflict monitoring and cognitive control. *Psychological review*, 108(3): 624.

Chen, H.; Feng, Y.; Liu, Z.; Yao, W.; Prabhakar, A.; Heinecke, S.; Ho, R.; Mui, P.; Savarese, S.; Xiong, C.; et al. 2024. Language models are hidden reasoners: Unlocking latent reasoning capabilities via self-rewarding. *arXiv preprint arXiv:2411.04282*.

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

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Creswell, A.; Shanahan, M.; and Higgins, I. 2022. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*.

Fu, Y.; Peng, H.; Sabharwal, A.; Clark, P.; and Khot, T. 2022. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*.

Geva, M.; Khashabi, D.; Segal, E.; Khot, T.; Roth, D.; and Berant, J. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9: 346–361.

Hao, S.; Sukhbaatar, S.; Su, D.; Li, X.; Hu, Z.; Weston, J.; and Tian, Y. 2024. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, 9118–9147. PMLR.

Koechlin, E.; Ody, C.; and Kouneiher, F. 2003. The architecture of cognitive control in the human prefrontal cortex. *Science*, 302(5648): 1181–1185.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213.

Ling, W.; Yogatama, D.; Dyer, C.; and Blunsom, P. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Liu, H.; Teng, Z.; Ning, R.; Liu, J.; Zhou, Q.; and Zhang, Y. 2023. Glore: Evaluating logical reasoning of large language models. *CoRR*.

Lu, P.; Mishra, S.; Xia, T.; Qiu, L.; Chang, K.-W.; Zhu, S.-C.; Tafjord, O.; Clark, P.; and Kalyan, A. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35: 2507–2521.

Nye, M.; Andreassen, A. J.; Gur-Ari, G.; Michalewski, H.; Austin, J.; Bieber, D.; Dohan, D.; Lewkowycz, A.; Bosma, M.; Luan, D.; et al. 2021. Show your work: Scratchpads for intermediate computation with language models.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744.

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

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Ruan, Y.; Band, N.; Maddison, C. J.; and Hashimoto, T. 2025. Reasoning to Learn from Latent Thoughts. *arXiv preprint arXiv:2503.18866*.

Saunshi, N.; Dikkala, N.; Li, Z.; Kumar, S.; and Reddi, S. J. 2025. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*.

Tang, J.; Dai, S.; Shi, T.; Xu, J.; Chen, X.; Chen, W.; Jian, W.; and Jiang, Y. 2025. Think Before Recommend: Unleashing the Latent Reasoning Power for Sequential Recommendation. *arXiv preprint arXiv:2503.22675*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023a. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36: 11809–11822.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Zhang, Y.; Zhang, B.; Li, Z.; Li, M.; Cheng, N.; Chen, M.; Wei, T.; Ma, J.; Wang, S.; and Xiao, J. 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)*, 1–5. IEEE.

Zhang, Z.; Zhang, A.; Li, M.; and Smola, A. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Zhou, D.; Schärli, N.; Hou, L.; Wei, J.; Scales, N.; Wang, X.; Schuurmans, D.; Cui, C.; Bousquet, O.; Le, Q.; et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.