LoLa: Looped Latent Reasoning via Hidden State Feedback

Anonymous authors
Paper under double-blind review

Abstract

Large reasoning models (LRMs) have achieved strong performance on reasoning tasks through chain-of-thought (CoT) generation, yet the resulting trajectories are often lengthy and redundant, leading to substantial inference overhead. Recent work has explored latent reasoning to improve efficiency, but existing methods typically either generate latent representations in a single forward pass without iterative refinement or build recurrence into the model during large scale pretraining. We propose **looped latent reasoning** (**LoLa**), a framework that combines a lightweight latent reasoner with a frozen base large language model (LLM). The reasoner maintains a latent state and iteratively updates it using hidden state feedback from the base LLM. The final state is projected into latent thought tokens that condition answer generation. We train the reasoner using supervised fine-tuning (SFT) followed by Group Relative Policy Optimization (GRPO). We evaluate LoLa with two base LLMs on five reasoning benchmarks. With DeepSeek-R1-Distill-Qwen-1.5B as the base model, LoLa outperforms existing efficient reasoning methods. With Qwen3-4B, LoLa improves pass@1 over its non-thinking mode from 54.07% to 58.64% and pass@4 from 65.78% to 72.36%, while also reducing inference latency. Once trained for a given base LLM, LoLa provides a plug-and-play reasoning mode that can be enabled on demand without modifying the base model, while explicit CoT remains available for harder problems. Code is available at `https://anonymous.4open.science/r/loopedlatent_iclr2027`.

1 Introduction

Chain-of-thought (CoT) reasoning improves language model performance by generating intermediate reasoning trajectories before final answers (Wei et al., 2022). Recent large reasoning models (LRMs), such as DeepSeek-R1 (Guo et al., 2025) and Qwen3 (Yang et al., 2025), further strengthen this capability through reinforcement learning (RL). However, their reasoning trajectories are often much longer than the final answers, and many tokens primarily serve as transitional text that maintains coherence. Prior work shows that models can often retain high answer accuracy even after a substantial portion of a reasoning trace is randomly removed (Xia et al., 2025; Jiang et al., 2026). This redundancy increases inference latency and memory consumption, contributing to the "overthinking" problem (Sui et al., 2025; Chen et al., 2024).

Several methods have been proposed to reduce this overhead. One line of work uses RL with length-related rewards to train models to produce shorter reasoning traces (Yi et al., 2025; Cheng et al., 2026; Luo et al., 2026; Hou et al., 2026), while another trains models to skip less important CoT tokens (Xia et al., 2025). These methods reduce the number of generated tokens, but reasoning remains autoregressive in the token space. Another line of work avoids explicit reasoning by prefilling a dummy thinking block (Ma et al., 2025) or using a built-in non-thinking mode (Yang et al., 2025). Although efficient, direct-answer modes can fall behind explicit reasoning when more inference computation is available. Existing approaches therefore either retain token-by-token reasoning or omit intermediate reasoning altogether. This raises a natural question: can the reasoning process be preserved in a more compact form without generating explicit reasoning text?

An increasing number of studies move intermediate reasoning into continuous latent spaces. Early work introduces special pause tokens (Goyal et al., 2024) or internalizes CoT through curriculum learning that gradually removes reasoning steps (Deng et al., 2024). Coconut (Hao et al., 2025)

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

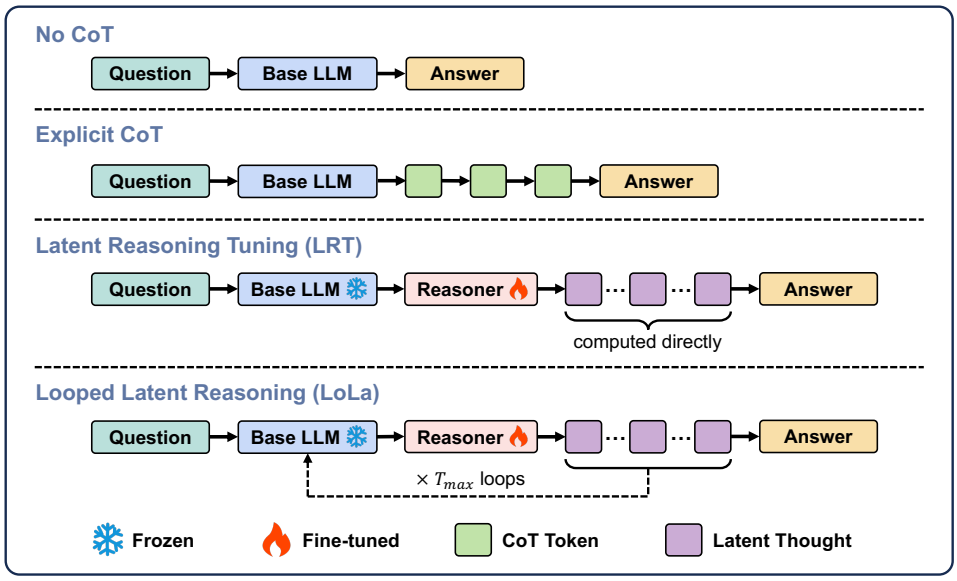


Figure 1: Comparison of reasoning paradigms: no CoT, explicit CoT, latent reasoning tuning (LRT), and our proposed looped latent reasoning (LoLa).

feeds the final hidden state back as the next input embedding, allowing the model to reason over continuous representations. CODI (Shen et al., 2025) and CCoT (Cheng & Durme, 2024) learn continuous reasoning representations through self-distillation or compression. More recently, latent reasoning tuning (LRT) (Jiang et al., 2026) introduces a lightweight reasoning network that maps the input to a latent representation of fixed length in a single forward pass, replacing autoregressive reasoning trajectory generation while leaving the base LLM unchanged. In parallel, looped models such as Ouro (Zhu et al., 2025) and HRM-Text (Wang et al., 2026) show that repeated computation over latent states can improve reasoning, but incorporate recurrence into the model architecture during pretraining. Recent reporting around GPT-6 Astra has brought renewed attention to recurrent-depth computation, where repeated internal processing increases the model's effective computational depth (Efrati et al., 2026).

These two directions leave a natural gap. LRT gains efficiency from latent reasoning, but produces its latent representation in a single forward pass and does not refine it using feedback from the base LLM. Looped models provide iterative computation, but existing approaches generally require the recurrent mechanism to be incorporated into the backbone and learned through large-scale pretraining. Building on both directions, we propose looped latent reasoning (LoLa), a framework that pairs a lightweight latent reasoner with a frozen base LLM. As illustrated in Figure 1, LoLa repeatedly exchanges hidden state information between the reasoner and the frozen base LLM for up to $T_{\max}$ rounds, rather than computing the latent representation in a single pass. The reasoner maintains an internal latent state, updates it using feedback from the base LLM at each round, and projects the final state into latent thought tokens that condition answer generation. Our primary contributions are as follows:

- We propose LoLa, a looped latent reasoning framework that iteratively refines a compact latent state using hidden-state feedback from a frozen base LLM, combining compact latent representations with iterative computation.
- LoLa is a modular reasoning component that leaves the base LLM unchanged, preserving its explicit CoT capability and allowing latent and explicit reasoning to coexist.
- Experiments with two backbone models on five benchmarks show that LoLa achieves the highest average performance among the compared efficient reasoning methods and Qwen3's non-thinking mode. In our efficiency evaluation, LoLa also achieves lower inference latency than the non-thinking mode.

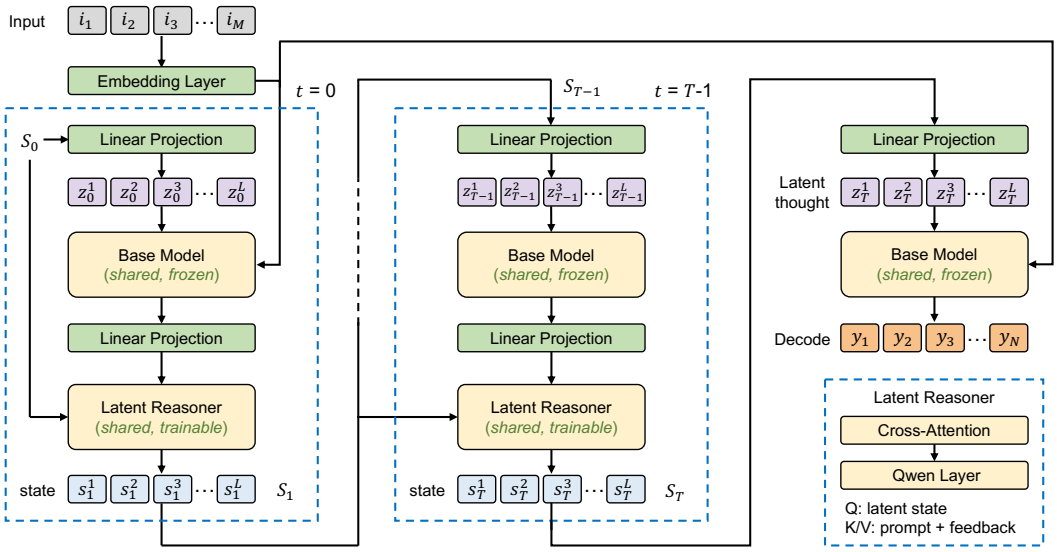


Figure 2: Architecture of LoLa. The main pipeline illustrates the looped latent state updates between a shared frozen base LLM and a shared trainable latent reasoner, followed by autoregressive answer generation conditioned on the final latent thought. The inset shows the latent reasoner with cross-attention adapters inserted before selected Qwen layers.

2 Looped Latent Reasoning

This section describes the LoLa framework. Given a prompt X and a frozen base LLM P_θ, LoLa learns a lightweight latent reasoner G_ϕ that produces a latent thought for answer generation without decoding explicit reasoning tokens. The reasoner maintains an internal latent state and refines it over multiple iterations. At each iteration, the current state is projected into latent thought vectors and passed to the base LLM. The base LLM returns hidden state feedback, which the reasoner uses to update the state. After T iterations, the final state is projected into a latent thought that conditions answer generation.

2.1 Architecture and Looped Updates

Architecture. LoLa has two main components: a frozen base LLM P_θ and a latent reasoner G_ϕ. The base LLM remains frozen throughout training and inference. Only the reasoner and its auxiliary modules are trained, including the cross-attention adapters, projection layers, initial-state parameter, loop embeddings, and boundary embeddings. Figure 2 shows the overall architecture.

The latent reasoner G_ϕ is a transformer initialized from a pretrained language model. It differs from the base LLM in two ways. First, it uses bidirectional self-attention, so all latent state vectors can attend to each other without causal masking. Second, at selected layers, it performs cross-attention over an external memory sequence that contains prompt information and feedback from the base LLM. In this cross-attention, the latent state vectors serve as queries, and keys and values come from the memory. Each cross-attention module uses a gated residual connection, where the gate is a learnable scalar initialized to a small value. This allows cross-attention to have limited effect early in training and lets the reasoner gradually learn to use the external information. Detailed architecture and cross-attention specifications are provided in Appendix A.

Algorithm 1 summarizes the full procedure, where t denotes the current iteration and T the total loop depth. After prompt encoding and latent state initialization, each iteration performs latent thought projection, base LLM feedback, and a looped state update.

Prompt Encoding and Latent State Initialization. Before the loop begins, the base LLM encodes the prompt X once. The prompt token embeddings E_X are passed through the frozen base LLM to obtain the final-layer hidden states H_X and the cached KV pairs KV_X. The hidden states are

Algorithm 1 LoLa inference with looped latent state updates

Input: Prompt X, base LLM P_θ, latent reasoner G_ϕ, learned initial state S_{init}, loop depth T, and scales γ_z, γ_s.
Output: Answer Y.
Begin
$E_X = \text{Embed}_\theta(X)$;
$(H_X, \text{KV}_X) = P_\theta(E_X)$; ▷ *Encode prompt and cache KV pairs*
$\mathcal{M}_X = f_{\text{p}}(H_X)$;
$S_0 = \text{RMSNorm}(S_{\text{init}})$; ▷ *Initialize latent state*
For $t = 0, \ldots, T-1$ **Do**
  $z_t = \gamma_z \cdot \text{RMSNorm}(f_{\text{out}}(S_t))$; ▷ *Project state to latent thought*
  $\widetilde{z}_t = [\, b_{\text{start}}, z_t, b_{\text{end}} \,]$;
  $B_t = \text{Hidden}_\theta(\widetilde{z}_t; \text{KV}_X)$; ▷ *Retain final layer states at latent positions*
  $F_t = f_{\text{fb}}(B_t)$;
  $\mathcal{M}_t = [\, \mathcal{M}_X, F_t \,]$; ▷ *Form memory for this loop*
  $\widehat{S}_{t+1} = G_\phi(S_t + E_{\text{loop}}[t]; \mathcal{M}_t)$; ▷ *Produce candidate state*
  $\Delta_t = \text{ClipRMS}\Big(\widehat{S}_{t+1} - S_t, \gamma_s\Big)$;
  $S_{t+1} = \text{RMSNorm}(S_t + \Delta_t)$;
End For
$z_T = \gamma_z \cdot \text{RMSNorm}(f_{\text{out}}(S_T))$; ▷ *Project final state*
$\widetilde{z}_T = [\, b_{\text{start}}, z_T, b_{\text{end}} \,]$;
$Y = \text{Decode}_\theta(E_X, \widetilde{z}_T)$; ▷ *Generate answer*
Return Y;
End

projected by f_{p} into the reasoner's dimension, forming the prompt memory $\mathcal{M}_X$. The KV cache is reused across all iterations so the base LLM does not need to re-encode the prompt.

The latent state $S_t \in \mathbb{R}^{L \times d_r}$ is the working representation maintained across iterations, where L is the number of latent vectors and d_r is the hidden dimension of the reasoner. The loop is initialized from a learnable parameter $S_{\text{init}} \in \mathbb{R}^{L \times d_r}$:

$$S_0 = \text{RMSNorm}(S_{\text{init}}). \tag{1}$$

Latent Thought Projection. At iteration t, the latent state S_t is projected to the base LLM's hidden dimension by an output projection f_{out}, followed by RMSNorm and scaling by γ_z:

$$z_t = \gamma_z \cdot \text{RMSNorm}(f_{\text{out}}(S_t)). \tag{2}$$

Here $z_t \in \mathbb{R}^{L \times d_b}$ is the latent thought at iteration t, where d_b is the hidden size of the base LLM. The scaling factor γ_z controls the magnitude of the latent thought. We wrap the latent thought with two learnable boundary embeddings before feeding it to the base LLM:

$$\widetilde{z}_t = [b_{\text{start}}, z_t, b_{\text{end}}].$$

Base LLM Feedback. The latent thought $\widetilde{z}_t$ is passed through the frozen base LLM, reusing the cached KV_X from prompt encoding. The base LLM processes $\widetilde{z}_t$ as a continuation of the prompt. We retain the hidden states from the final layer that correspond to the L latent thought positions, excluding the two boundary positions, and denote them by $B_t \in \mathbb{R}^{L \times d_b}$. These hidden states provide position-wise feedback on the current latent thought. A feedback projection f_{fb} maps them into the reasoner's dimension, and the projected feedback is concatenated with the prompt memory:

$$\mathcal{M}_t = [\mathcal{M}_X, f_{\text{fb}}(B_t)].$$

If the prompt length is M, then $\mathcal{M}_t \in \mathbb{R}^{(M+L) \times d_r}$. Because z_t changes across iterations, the base LLM provides feedback specific to the current latent thought at each iteration.

Looped State Update. The reasoner takes $S_t + E_{\text{loop}}[t]$ as input, where $E_{\text{loop}}[t]$ is a learned per-loop embedding that allows the reasoner to distinguish different iterations. It attends to $\mathcal{M}_t$ through cross-attention and produces a candidate state:

$$\widehat{S}_{t+1} = G_\phi\left(S_t + E_{\text{loop}}[t]; \mathcal{M}_t\right). \tag{3}$$

The residual update is clipped before being added to the current state:

$$\Delta_t = \mathrm{ClipRMS}\left(\widehat{S}_{t+1} - S_t, \gamma_s\right), \qquad S_{t+1} = \mathrm{RMSNorm}\left(S_t + \Delta_t\right). \tag{4}$$

For the update at latent position i, $u_{t,i} = \widehat{S}_{t+1,i} - S_{t,i}$, RMS-based clipping is defined as

$$\mathrm{ClipRMS}(u_{t,i}, \gamma_s) = u_{t,i} \cdot \min\left(1, \frac{\gamma_s}{\mathrm{RMS}(u_{t,i}) + \epsilon}\right). \tag{5}$$

Here, $\mathrm{RMS}(\cdot)$ denotes the root mean square and ϵ is a small constant for numerical stability. The clipping operation leaves an update unchanged when its RMS does not exceed γ_s and rescales it otherwise. The subsequent RMSNorm stabilizes the state scale across iterations.

After T iterations, the final state S_T is projected to obtain the final latent thought z_T, which is wrapped with boundary embeddings as $\widetilde{z}_T$. The base LLM then generates the answer Y autoregressively, conditioned on $[E_X, \widetilde{z}_T]$.

2.2 Training Pipeline

SFT. Each training example consists of a prompt X and a target solution sequence $Y = [y_1, \ldots, y_N]$ from OpenR1-Math-220k. LoLa first runs the looped update procedure to obtain the final latent thought $\widetilde{z}_T$. The latent thought receives no direct supervision or representation-level alignment. We instead optimize the average negative log likelihood of the target solution tokens:

$$\mathcal{L}_{\mathrm{SFT}}(\phi) = -\frac{1}{N}\sum_{j=1}^{N} \log P_\theta\left(y_j \mid E_X, \widetilde{z}_T, y_{<j}\right). \tag{6}$$

Gradients are propagated through the frozen base LLM during feedback and answer generation to optimize the reasoner and its auxiliary modules.

GRPO. The SFT-trained model is used as the initial policy. In the GRPO stage, we use a fixed loop depth $T_{\mathrm{RL}} = 3$, selected based on SFT validation performance and used throughout GRPO training and the main evaluation. For each prompt, the policy computes the final latent thought $\widetilde{z}_{T_{\mathrm{RL}}}$ and samples multiple candidate responses from the frozen base LLM. Each response receives a reward based on final-answer correctness. We normalize rewards within candidates from the same prompt and optimize the latent reasoner following GRPO (Shao et al., 2024). We define the asymmetrically clipped token-level policy ratio as

$$\bar{\rho}_{k,j} = \mathrm{clip}\left(\rho_{k,j}, 1 - \epsilon_{\mathrm{low}}, 1 + \epsilon_{\mathrm{high}}\right). \tag{7}$$

The resulting RL objective is

$$\mathcal{L}_{\mathrm{RL}}(\phi) = -\mathbb{E}_k\left[\frac{1}{|Y_k|}\sum_{j=1}^{|Y_k|} \min\left(\rho_{k,j}A_k,\ \bar{\rho}_{k,j}A_k\right)\right] + \beta_{\mathrm{KL}}\mathcal{L}_{\mathrm{KL}}(\phi). \tag{8}$$

Here, A_k is the group-normalized advantage of the k-th candidate, $\rho_{k,j}$ is the ratio between the current and sampling policies at its j-th response token, and $|Y_k|$ is the number of valid response tokens. We use different lower and upper clipping ranges following the DAPO-style asymmetric clipping design (Yu et al., 2025). The coefficient β_{KL} controls the KL penalty, for which the SFT checkpoint serves as the fixed reference policy.

3 Experiments

3.1 Settings

Models. We evaluate LoLa with DeepSeek-R1-Distill-Qwen-1.5B (Guo et al., 2025) and Qwen3-4B (Yang et al., 2025) as base LLMs. DeepSeek-R1-Distill-Qwen-1.5B is optimized for reasoning and is commonly used in efficient reasoning research. Qwen3 natively supports thinking and non-thinking modes through its chat template, allowing us to compare LoLa with its non-thinking mode using the same base LLM. Following LRT (Jiang et al., 2026), we initialize the latent reasoner backbone from Qwen3-Embedding-0.6B (Zhang et al., 2025) and set the latent state length to 256.

Datasets. We use OpenR1-Math-220k (Hugging Face, 2025) for SFT, where we use the provided solution field as the target sequence, and the training splits of GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021) for GRPO. For evaluation, we use five benchmarks spanning mathematical, logical, and scientific reasoning. GSM8K (Cobbe et al., 2021) evaluates multi-step arithmetic reasoning on grade-school word problems. MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024) is a challenging subset of competition-level mathematics problems. AMC (Mathematical Association of America, 2023) focuses on non-routine mathematical problem solving. LSAT (Zhong et al., 2024) evaluates logical reasoning and reading comprehension through questions involving complex arguments. GPQA (**?**) comprises graduate-level questions in physics, chemistry, and biology that require specialized domain knowledge.

Baselines. On DeepSeek-R1-Distill-Qwen-1.5B, we compare LoLa with the base model and several efficient reasoning methods. NoThinking (Ma et al., 2025) avoids explicit reasoning by directly generating answers. ShorterBetter (Yi et al., 2025) uses RL with a sample optimal length reward to encourage more efficient reasoning traces. LC-R1 (Cheng et al., 2026) uses length and compression rewards to reduce redundant reasoning. We further compare with LRT (Jiang et al., 2026), which uses a single forward pass through a latent reasoning network to generate fixed-length latent representations. For Qwen3-4B, we compare LoLa with the non-thinking mode and LRT. Additional experimental details are provided in Appendix B.

3.2 Main Results

Table 1: Accuracy (%) of different reasoning methods on DeepSeek-R1-Distill-Qwen-1.5B.

Method	Mathematical reasoning			Logic and science		Average
	AMC	MATH-500	GSM8K	LSAT	GPQA	
Baseline	42.88	67.50	79.10	20.98	28.16	47.72
NoThinking	40.25	66.70	75.00	22.72	25.88	46.11
ShorterBetter	36.31	55.76	60.78	18.32	28.38	39.91
LC-R1	**44.87**	68.00	78.98	20.22	30.56	48.53
LRT	42.50	**68.50**	78.95	22.39	30.55	48.58
LoLa (Ours)	44.58	67.00	**80.36**	**24.35**	**30.81**	**49.42**

Results on DeepSeek-R1-Distill-Qwen-1.5B. Table 1 reports the results on DeepSeek-R1-Distill-Qwen-1.5B. Results for the baseline methods are taken from LRT (Jiang et al., 2026), while LoLa is evaluated under the same budget forcing setting, with a maximum generation length of 1,024 tokens. LoLa uses a loop depth of $T = 3$ in the main experiments, and results for other depths are reported in Appendix C.1. LoLa achieves the highest average accuracy of 49.42% across the five benchmarks.

On the mathematical reasoning benchmarks, LoLa achieves the highest accuracy on GSM8K at 80.36%. On AMC, it reaches 44.58%, close to the best result of 44.87% achieved by LC-R1. On MATH-500, LoLa scores slightly below LRT but remains comparable to the other methods. On the logic and science benchmarks, LoLa achieves the highest accuracy on both LSAT and GPQA. Compared with the best competing method on each benchmark, LoLa improves LSAT accuracy from 22.72% with NoThinking to 24.35% and GPQA accuracy from 30.56% with LC-R1 to 30.81%. Overall, LoLa remains competitive on mathematical reasoning while achieving the best results on both LSAT and GPQA.

Results on Qwen3-4B. Table 2 compares LoLa with the Qwen3 non-thinking mode and LRT on Qwen3-4B. Here, base denotes the Qwen3 non-thinking mode, and @k denotes pass@k performance. At pass@1, LoLa achieves the highest average score of 58.64%, exceeding base@1 and LRT@1 by 4.57 and 3.60 percentage points, respectively. LoLa obtains the highest scores on MATH-500, GSM8K, and LSAT, while base performs better on AMC and GPQA.

At pass@4, LoLa achieves the highest average score of 72.36%, exceeding base@4 by 6.58 percentage points and LRT@4 by 0.76 percentage points. LoLa obtains the highest scores on AMC, GSM8K, and LSAT, while LRT performs better on MATH-500 and GPQA. The gains over base@4 are particularly

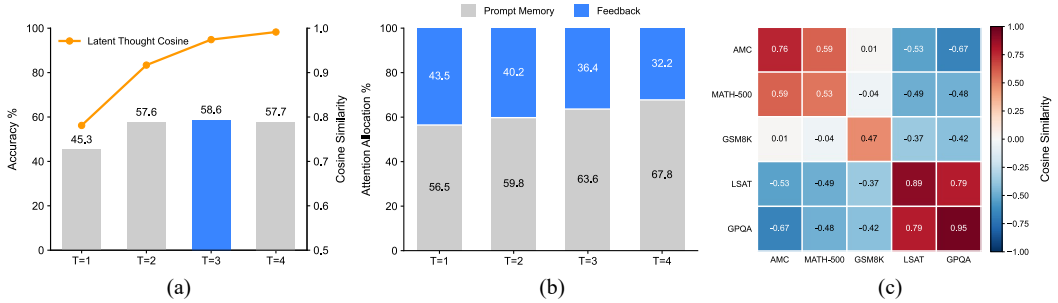
Table 2: Performance comparison of the non-thinking mode, LRT, and LoLa on Qwen3-4B.

Method	Mathematical reasoning			Logic and science		Average
	AMC	MATH-500	GSM8K	LSAT	GPQA	
base@1	**47.25**	70.70	75.08	32.50	**44.82**	54.07
LRT@1	46.25	72.60	88.51	28.59	39.27	55.04
LoLa@1	45.78	**75.20**	**91.51**	**44.35**	36.36	**58.64**
base@4	54.00	80.00	88.10	42.17	64.65	65.78
LRT@4	54.00	**84.80**	95.07	56.96	**67.17**	71.60
LoLa@4	**61.45**	82.00	**95.98**	**61.74**	60.61	**72.36**

large on GSM8K and LSAT, where LoLa improves the scores from 88.10% to 95.98% and from 42.17% to 61.74%, respectively. LoLa provides a stronger alternative to the native non-thinking mode under both pass@1 and pass@4 evaluation, with larger average gains when multiple samples are used. A detailed comparison of inference efficiency is provided in Appendix C.2.

3.3 Analysis and Ablation Study

We further analyze LoLa on Qwen3-4B along four dimensions: iterative refinement, the use of prompt and feedback memory, variation in the learned latent thoughts across tasks, and the contributions of feedback from the base model and GRPO training.



(a) (b) (c)

Figure 3: Analysis of the looped latent reasoning mechanism. (a) Accuracy and cosine similarity between adjacent latent thoughts at different loop depths. (b) Cross-attention allocation to prompt memory and feedback memory across loops. (c) Average cosine similarity of latent thought representations within and across benchmarks.

Latent Thought Convergence. Figure 3(a) shows how loop depth affects average accuracy and the cosine similarity between adjacent latent thoughts. Increasing the depth from $T = 1$ to $T = 2$ yields a substantial accuracy gain, from 45.26% to 57.60%. The highest average accuracy of 58.64% is achieved at $T = 3$, while increasing the depth to $T = 4$ slightly reduces it to 57.74%. Meanwhile, the cosine similarity between adjacent latent thoughts increases from 0.78 to 0.99 as the loop depth grows, suggesting that the latent state undergoes larger changes in the early loops and gradually stabilizes in later loops. This behavior is consistent with findings from recurrent and looped language models, where additional iterations can improve performance but the gains eventually saturate and may decline beyond an appropriate depth (Geiping et al., 2025; Zhu et al., 2025; Yang et al., 2026).

Attention Dynamics. Figure 3(b) shows how the reasoner distributes cross-attention between prompt memory and feedback memory. The share assigned to prompt memory increases from 56.47% in the first loop to 67.76% in the fourth loop, while the share assigned to feedback decreases from 43.53% to 32.24%. This shift suggests that the early updates draw more heavily on feedback from the base LLM, whereas later updates place greater weight on the prompt information.

Representations Across Tasks. Figure 3(c) shows the cosine similarity of latent thought representations within and across benchmarks. For each example, we average the latent thought at $T = 3$ over its latent positions, subtract the global mean from the resulting representation, and compute the average pairwise cosine similarities within and across benchmarks. The heatmap reveals clear differences between the two benchmark groups. AMC and MATH-500 have a positive similarity of 0.59, consistent with both being competition mathematics benchmarks. LSAT and GPQA also show a high similarity of 0.79. In contrast, similarities between the two groups are mostly negative: AMC has similarities of -0.53 with LSAT and -0.67 with GPQA. GSM8K shows a different pattern, with similarities near zero to AMC and MATH-500 and negative similarities to LSAT and GPQA. These results suggest that LoLa does not produce uniform latent representations across benchmarks; instead, its latent thoughts retain structure associated with different types of reasoning tasks.

Table 3: Ablation of base LLM feedback at loop depth T=3.

Loop configuration	Mathematical reasoning			Logic and science		Average
	AMC	MATH-500	GSM8K	LSAT	GPQA	
Reasoner only	40.96	74.40	91.36	40.87	35.35	56.59
Reasoner + base LLM	**45.78**	**75.20**	**91.51**	**44.35**	**36.36**	**58.64**

Ablation of Base LLM Feedback. Table 3 examines the effect of feedback from the frozen base LLM. When only the reasoner is included in the loop, the latent state is iteratively updated using the prompt memory, but the latent thought is not passed through the base LLM and no feedback memory is formed. In the full LoLa setting, each latent thought is processed by the base LLM, and the resulting hidden states are projected to form the feedback memory used in the next state update. Including the base LLM in the loop increases the average accuracy from 56.59% to 58.64%, a gain of 2.05 percentage points, with improvements on all five benchmarks. The largest gains occur on AMC and LSAT, where accuracy increases by 4.82 and 3.48 percentage points, respectively. The improvements on MATH-500, GSM8K, and GPQA are smaller but remain positive. These results indicate that feedback from the base LLM contributes to more effective latent state refinement.

Table 4: Accuracy (%) of LoLa after different training stages.

Training stage	Mathematical reasoning			Logic and science		Average
	AMC	MATH-500	GSM8K	LSAT	GPQA	
SFT	42.17	68.00	78.77	31.74	32.32	50.60
SFT + GRPO	**45.78**	**75.20**	**91.51**	**44.35**	**36.36**	**58.64**

Ablation of Training Stages. Table 4 compares LoLa after SFT alone and after SFT followed by GRPO. Adding the GRPO stage increases the average accuracy from 50.60% to 58.64%, an improvement of 8.04 percentage points, with gains on all five benchmarks. The largest improvements are observed on GSM8K and LSAT, where accuracy increases by 12.74 and 12.61 percentage points, respectively. Performance also improves on AMC, MATH-500, and GPQA. These results show that SFT establishes the initial latent reasoning capability, while GRPO further improves performance under the main evaluation setting.

4 Related Work

Compressing Explicit CoT. Several studies reduce inference cost by shortening explicit CoT traces. C3oT (Kang et al., 2025) uses a compressor to transform long CoT traces into shorter ones and jointly trains the model on both forms through conditioned prompts. TokenSkip (Xia et al., 2025) estimates token importance to construct compressed training traces at different ratios, then fine-tunes the model to skip less informative tokens while retaining critical reasoning content. Other work uses reinforcement learning to shorten reasoning traces. O1-Pruner (Luo et al., 2026) estimates problem-specific reference performance through pre-sampling and applies length-harmonizing fine-tuning to

reduce reasoning overhead while preserving accuracy. ThinkPrune (Hou et al., 2026) uses GRPO with length clipping and progressively tighter token limits. ShorterBetter (Yi et al., 2025) defines the sample optimal length as the shortest correct response among multiple rollouts and uses it as a reward signal for efficient reasoning. LC-R1 (Cheng et al., 2026) targets invalid thinking after the correct answer has already been reached and combines a Length Reward with a Compress Reward to remove redundant verification while preserving necessary reasoning.

Latent Reasoning. Latent reasoning methods reduce or avoid explicit CoT generation by moving intermediate computation into hidden states or continuous representations. Pause Tokens (Goyal et al., 2024) append learnable pause tokens to the input, allowing the model to perform additional computation before producing an output. Stepwise Internalization (Deng et al., 2024) gradually removes CoT tokens during fine-tuning, forcing the model to internalize the omitted reasoning steps. Coconut (Hao et al., 2025) feeds the final hidden state back as the next input embedding and uses a curriculum that progressively replaces explicit reasoning steps with continuous thoughts. CCoT (Cheng & Durme, 2024) trains auxiliary modules to generate variable-length continuous contemplation tokens that approximate compressed representations of explicit CoT. CODI (Shen et al., 2025) jointly trains an explicit-CoT teacher task and an implicit-CoT student task within the same model, using hidden state alignment to distill reasoning into continuous space. PCCoT (Wu et al., 2025) addresses the sequential dependency between continuous thought tokens by updating them in parallel through Jacobi iteration. LRT (Jiang et al., 2026) adopts a modular design in which a lightweight reasoning network generates a compact latent trajectory in a single forward pass while the base LLM remains frozen.

Looped Language Models. Looped models repeatedly apply shared computation blocks, increasing effective depth without proportionally increasing the number of parameters. Universal Transformer (Dehghani et al., 2019) applies shared self-attention blocks recurrently over depth and uses adaptive computation time for position-wise halting. Huginn (Geiping et al., 2025) trains a recurrent-depth transformer from scratch with a randomized number of loop iterations, allowing test-time computation to scale through additional latent recurrence. Ouro (Zhu et al., 2025) incorporates iterative latent computation and entropy-regularized adaptive depth during pretraining, enabling the model to allocate different numbers of recurrent steps across inputs. HRM-Text (Wang et al., 2026) uses hierarchical dual-timescale recurrence, with fast execution updates and slower strategic updates, together with stabilization techniques for deep recurrent training. LoopCoder-v2 (Yang et al., 2026) studies parallel loop transformers for code and finds that a two-loop model improves broadly over the non-looped baseline, whereas additional loops yield diminishing or negative returns.

5 Conclusion

We propose LoLa, a looped latent reasoning framework that combines a frozen base LLM with a lightweight latent reasoner. Rather than generating explicit CoT tokens, LoLa maintains a latent state and iteratively refines it using feedback from the base LLM, enabling latent reasoning without updating the parameters of the base model. Experiments with DeepSeek-R1-Distill-Qwen-1.5B and Qwen3-4B show that LoLa achieves the highest average performance among the compared methods on the five reasoning benchmarks. Our analysis shows that iterative refinement is most effective at a moderate loop depth, feedback from the base LLM consistently improves performance, and GRPO further improves the model obtained after SFT. These results suggest that compact latent loops can provide effective intermediate computation under limited inference budgets, providing an alternative to explicit CoT reasoning under inference constraints.

AI Use Statement

In this work, we used generative artificial intelligence (AI) tools only to edit the manuscript for readability and grammar. We did not use generative AI tools to develop the research methodology, formulate mathematical claims, implement the proposed method, design experiments, or interpret the results. All AI-assisted edits were reviewed by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Reproducibility Statement

We provide detailed descriptions of the LoLa architecture and training procedure in Sections 2.1 and 2.2. Additional architecture specifications, training hyperparameters, and inference settings are provided in Appendix A and Appendix B. Code is available at `https://anonymous.4open.science/r/loopedlatent_iclr2027`.

References

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

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

Zhengxiang Cheng, Dongping Chen, Mingyang Fu, and Tianyi Zhou. Optimizing length compression in large reasoning models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3232–3250, 2026.

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

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *7th International Conference on Learning Representations*, 2019.

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

Amir Efrati, Stephanie Palazzolo, and Rocket Drew. Openai technique in 'astra' model sparks security concerns. The Information, September 2026.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems 38*, 2025.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*, 2024.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1*, 2021.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *Trans. Mach. Learn. Res.*, 2026.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL `https://github.com/huggingface/open-r1`.

Cong Jiang, Xiaofeng Zhang, Fangzhi Zhu, XiaoWei Chen, Junxiong Zhu, and Zheng Zhang. Rethinking llm reasoning: From explicit trajectories to latent representations. In *The Fourteenth International Conference on Learning Representations*, 2026.

Yu Kang, Xianghui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. In *Thirty-Ninth AAAI Conference on Artificial Intelligence*, pp. 24312–24320, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

Haotian Luo, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Xiaochun Cao, Dacheng Tao, Naiqiang Tan, and Li Shen. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. In *Findings of the Association for Computational Linguistics*, pp. 14242–14257, 2026.

Wenjie Ma, Jingxuan He, Charlie Snell, Tyler Griggs, Sewon Min, and Matei Zaharia. Reasoning models can be effective without thinking. *arXiv preprint arXiv:2504.09858*, 2025.

Mathematical Association of America. American mathematics competitions, 2023.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel J. Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. *Trans. Mach. Learn. Res.*, 2025.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers Reinforcement Learning, 2020. URL `https://github.com/huggingface/trl`.

Guan Wang, Changling Liu, Chenyu Wang, Cai Zhou, Yuhao Sun, Yifei Wu, Shuai Zhen, Luca Scimeca, and Yasin Abbasi Yadkori. Hrm-text: Efficient pretraining beyond scaling. *arXiv preprint arXiv:2605.20613*, 2026.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35*, 2022.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020.

Haoyi Wu, Zhihao Teng, and Kewei Tu. Parallel continuous chain-of-thought with jacobi iteration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 914–926, 2025.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 3351–3363, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Jian Yang, Shawn Guo, Wei Zhang, Tianyu Zheng, Yaxin Du, Haau-Sing Li, Jiajun Wu, Yue Song, Yan Xing, Qingsong Cai, et al. Loopcoder-v2: Only loop once for efficient test-time computation scaling. *arXiv preprint arXiv:2606.18023*, 2026.

Jingyang Yi, Jiazheng Wang, and Sida Li. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning. In *Advances in Neural Information Processing Systems 38*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems 38*, 2025.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics*, pp. 2299–2314, 2024.

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

A Architecture Details

The latent reasoner retains the 28 transformer layers and the final normalization layer of its backbone, with hidden dimension $d_r = 1024$. Its token embedding layer and output head are not used because the reasoner directly processes continuous latent states. We replace the causal attention mask with a bidirectional mask and retain the original rotary position embeddings, using position indices $0, \ldots, L-1$ for the $L = 256$ latent vectors. The same transformer parameters are shared across all loops.

Cross-attention adapters are inserted before layers $0, 4, 8, 12, 16, 20$, and 24. For an adapter inserted before layer ℓ, let $U \in \mathbb{R}^{L \times d_r}$ denote its input latent representation. The latent representation and memory $\mathcal{M}_t$ are normalized separately before being projected to queries, keys, and values:

$$\begin{aligned} Q &= \mathrm{RMSNorm}_{q,\ell}(U) W_\ell^Q, \\ K &= \mathrm{RMSNorm}_{kv,\ell}(\mathcal{M}_t) W_\ell^K, \\ V &= \mathrm{RMSNorm}_{kv,\ell}(\mathcal{M}_t) W_\ell^V. \end{aligned} \tag{9}$$

Here, $W_\ell^Q, W_\ell^K, W_\ell^V \in \mathbb{R}^{d_r \times d_r}$ are the learned query, key, and value projection matrices of the adapter at layer ℓ. For each of the H attention heads, the cross-attention output is

$$C_h = \mathrm{softmax}\left(\frac{Q_h K_h^\top}{\sqrt{d_h}} + A_t\right) V_h, \qquad h = 1, \ldots, H, \tag{10}$$

where $d_h = d_r / H$ is the head dimension. The additive mask A_t assigns $-\infty$ to padded prompt positions and zero elsewhere, preventing attention to padding while preserving access to all valid prompt and feedback positions.

The outputs of the H attention heads are concatenated and transformed by a learned output projection $W_\ell^O \in \mathbb{R}^{d_r \times d_r}$. Each adapter then applies a gated residual update:

$$U^+ = U + g_\ell \, \mathrm{Concat}(C_1, \ldots, C_H) W_\ell^O, \qquad g_\ell = \sigma(a_\ell). \tag{11}$$

The scalar gate is learned independently for each adapter. We initialize $a_\ell = \mathrm{logit}(0.1)$ so that $g_\ell = 0.1$ at initialization. The cross-attention adapters use the same number of attention heads as the reasoner backbone, and cross-attention dropout is set to zero.

The prompt and feedback projections f_{p} and f_{fb} are separate bias-free linear layers from the base-model dimension d_b to d_r, while the latent-thought projection f_{out} is a bias-free linear layer from d_r to d_b. The latent output scale is $\gamma_z = 0.03$, and the per-position RMS limit on state updates is $\gamma_s = 0.5$. All newly added RMSNorm layers use $\epsilon = 10^{-6}$. The learnable initial-state parameter S_{init}, loop embeddings, and boundary embeddings are initialized from $\mathcal{N}(0, 0.02^2)$.

B Experiment Details

Training Setup. We train LoLa on four NVIDIA A100 GPUs. GRPO training is implemented with the open-source TRL library (von Werra et al., 2020). For both base LLMs, the latent reasoner is initialized from Qwen3-Embedding-0.6B (Zhang et al., 2025) and uses 256 latent vectors (Jiang et al., 2026), while each base LLM remains frozen throughout training. The prompt representations and corresponding KV caches are computed without gradient tracking. During each feedback pass, the computation remains differentiable with respect to z_t, allowing gradients to propagate through the feedback path to the latent reasoner.

In the SFT stage, we train on OpenR1-Math-220k for two epochs with an effective batch size of 128 and a learning rate of 1×10^{-4}. We use a loop depth of $T = 4$ and apply supervision only to the autoregressive generation of the target solution conditioned on the final latent thought. The latent thought itself does not receive an additional supervision signal. We use a cosine learning-rate schedule with 100 warmup steps.

In the GRPO stage, each optimization step uses 16 questions with eight rollouts per question, resulting in an effective rollout batch size of 128. The prompt and rollout lengths are each limited to 1,024 tokens, and rollouts are sampled with a temperature of 1.0. The reward depends only on final-answer

correctness. We use the SFT model as the fixed reference policy, with a KL coefficient of 1×10^{-3} and lower and upper clipping bounds of 0.2 and 0.28, respectively. For Qwen3-4B, we train on the GSM8K training set for 50 steps with a learning rate of 3×10^{-5}. For DeepSeek-R1-Distill-Qwen-1.5B, we first train on the GSM8K training set for 50 steps with the same learning rate, followed by 25 steps on the MATH training set with a learning rate of 1×10^{-5}.

Inference Setup. We follow the inference setup of LRT (Jiang et al., 2026) and use the same settings for all compared methods. We evaluate LoLa on a single NVIDIA A100 GPU using Hugging Face Transformers (Wolf et al., 2020), with a generation temperature of 0.6, top-p sampling with $p = 0.95$, and a generation budget of 1,024 tokens. Following LRT, we adopt the budget-forcing procedure introduced in S1 (Muennighoff et al., 2025). This common setup enables direct comparison across methods.

C Additional Experiments and Analysis

C.1 Effect of Loop Depth

Table 5: Performance of LoLa at different loop depths T on Qwen3-4B.

T	Mathematical reasoning			Logic and science		Average
	AMC	MATH-500	GSM8K	LSAT	GPQA	
1	34.94	64.20	88.93	10.43	27.78	45.26
2	42.17	**75.60**	**92.27**	39.57	38.38	57.60
3	45.78	75.20	91.51	**44.35**	36.36	**58.64**
4	**46.99**	73.80	91.21	37.83	**38.89**	57.74

We evaluate LoLa on Qwen3-4B with loop depths from $T = 1$ to $T = 4$, keeping all other inference settings fixed. As shown in Table 5, increasing the loop depth from $T = 1$ to $T = 2$ raises the average accuracy from 45.26% to 57.60%, with improvements on all five benchmarks. Increasing the depth to $T = 3$ yields a smaller additional gain and the highest average accuracy of 58.64%, while $T = 4$ slightly reduces the average to 57.74%. The optimal depth varies across tasks: MATH-500 and GSM8K achieve their highest accuracy at $T = 2$, LSAT at $T = 3$, and AMC and GPQA at $T = 4$. Thus, additional loops do not benefit all benchmarks uniformly. We use $T = 3$ in the main evaluation because it achieves the highest average accuracy across the five benchmarks.

C.2 Inference Efficiency Comparison

Table 6: Inference efficiency on Qwen3-4B. The terms *non-thinking* and *thinking* refer to the standard and reasoning modes, respectively. For LoLa, the value before the slash is the generation throughput, while the value marked with † additionally counts latent thought positions.

Method	Latency (sec/question) ↓	Throughput (tokens/sec) ↑	Peak memory (MB) ↓
thinking	162.85	29.98	10042
non-thinking	28.16	**31.19**	**8488**
LoLa	**24.19**	30.49 / 41.07†	9140

We compare the inference efficiency of LoLa with the thinking and non-thinking modes of Qwen3-4B. Following LRT (Jiang et al., 2026), we use the same 64 randomly sampled problems from MATH-500 and measure end-to-end latency, token throughput, and peak GPU memory on a single NVIDIA A100 GPU with batch size 1. All methods use the same sampling configuration and a maximum generation length of 8,192 tokens per question. LoLa uses a loop depth of $T = 3$, and its latency includes both latent-state refinement and answer generation.

As shown in Table 6, LoLa achieves the lowest average latency at 24.19 seconds per question, compared with 28.16 seconds for the non-thinking mode and 162.85 seconds for the thinking mode.

Its generation throughput is 30.49 tokens per second, close to the 31.19 tokens per second achieved by the non-thinking mode. The lower latency mainly results from generating fewer autoregressive tokens, while latent-state refinement does not require token-by-token decoding.

Following the accounting convention used by LRT, we also report an effective throughput that counts the 256 latent thought positions together with the generated tokens. Because these latent positions are processed in parallel, the effective throughput is 41.07 tokens per second, compared with a generation throughput of 30.49 tokens per second. This adjusted value reflects the additional latent computation and should not be interpreted as autoregressive decoding throughput.

LoLa uses 9,140 MB of peak GPU memory, which lies between the non-thinking mode at 8,488 MB and the thinking mode at 10,042 MB. The additional memory relative to the non-thinking mode arises from the latent reasoner and looped feedback computation. Overall, LoLa reduces latency relative to both Qwen3 modes while requiring moderately more memory than non-thinking inference.

C.3 Evaluation Variability

Table 7: Mean accuracy and standard deviation over five evaluation runs. Accuracy is reported in percent, and standard deviation in percentage points.

Model	Metric	Benchmarks				
		AMC	MATH-500	GSM8K	LSAT	GPQA
DeepSeek-R1-Distill-Qwen-1.5B	Mean accuracy	43.07	67.65	79.64	23.26	29.39
	Standard deviation	3.98	1.68	0.72	1.35	1.45
Qwen3-4B	Mean accuracy	47.89	74.10	91.93	39.78	38.38
	Standard deviation	1.52	0.68	0.31	0.75	2.89

Because decoding is stochastic, evaluation results may vary across runs. The main tables report results from a single run. To quantify this variation, we evaluate LoLa five times with different random seeds while keeping the model checkpoint and all other inference settings fixed. Table 7 reports the mean accuracy and standard deviation across the five runs.

Variability across the five evaluation runs differs across benchmarks. For DeepSeek-R1-Distill-Qwen-1.5B, the standard deviation ranges from 0.72 percentage points on GSM8K to 3.98 points on AMC. For Qwen3-4B, it ranges from 0.31 points on GSM8K to 2.89 points on GPQA. Eight of the ten model and benchmark combinations have a standard deviation below two percentage points, while larger variation occurs on AMC with DeepSeek-R1-Distill-Qwen-1.5B and GPQA with Qwen3-4B. The results show limited variation in most settings and provide a reference for interpreting the single run results in the main tables.