THINKBRIDGE: ELICITING LATENT REASONING FROM LARGE REASONING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Explicit chain-of-thought (CoT) helps large language models (LLMs) solve problems, but generating long CoTs token by token incurs substantial inference latency. Existing latent reasoning methods fine-tune LLMs to carry intermediate computation in continuous representations. Reasoning-focused post-training has produced large reasoning models (LRMs), a class of LLMs with strong explicit reasoning capabilities. In preliminary mathematical reasoning experiments, we examine conventional supervised fine-tuning and existing latent reasoning methods on LRMs and find marked accuracy drops in some settings relative to the unmodified models. We also find that the original LRMs provide complementary successful responses in thinking and non-thinking modes. These observations motivate ThinkBridge, which keeps the LRM frozen and trains a lightweight reasoner using the model's own successful responses. The reasoner constructs compact latent states that guide the LRM to respond without first generating an explicit CoT. Training combines three objectives: supervision on correct native responses supports accurate answers; on-policy distillation aligns latent-conditioned and full-CoT-conditioned response predictions, training latent states to guide responses in place of CoT; and contrastive learning encourages question-specific latent states that support reasoning about the current question. We train ThinkBridge on GSM8K-Aug-NL. Across five mathematical benchmarks, it achieves the highest mean accuracy among the evaluated latent reasoning methods at 0.6B and 4B, with mean times to first response token (TTFT) of 45.50 and 56.33 ms, respectively. It also outperforms these baselines in multi-turn accuracy without dialogue-specific training.

1 INTRODUCTION

Explicit chain-of-thought (CoT) helps large language models (LLMs) solve problems by expressing intermediate reasoning steps in text (Wei et al., 2022; Kojima et al., 2022). Generating a long CoT token by token, however, delays the response and increases inference cost (Sui et al., 2025; Han et al., 2025). Latent reasoning offers an alternative way to reduce this cost. These methods carry intermediate computation in continuous representations without generating a complete textual reasoning trace (Hao et al., 2025; Shen et al., 2025).

Prior studies have fine-tuned LLMs to reason in continuous space, as illustrated in Figure 1(a). Stepwise internalization, Coconut, and CODI follow this approach (Deng et al., 2024; Hao et al., 2025; Shen et al., 2025). Reasoning-focused post-training has produced a class of LLMs with strong explicit reasoning capabilities, known as large reasoning models (LRMs) (Muennighoff et al., 2025; Yang et al., 2025). For example, DeepSeek-R1 (Guo et al., 2025) and Qwen3 (Yang et al., 2025) achieve strong performance on mathematics and coding tasks.

We conduct preliminary experiments in mathematical reasoning to examine how conventional supervised fine-tuning and existing latent reasoning methods perform on LRMs. We find marked accuracy drops in some settings relative to the unmodified models. Models fine-tuned on the dataset's original CoTs also generate markedly shorter text, with a style that differs from the models' native outputs (Section 2.1). We also find that the original LRMs solve different, partly overlapping sets of questions in thinking and non-thinking modes. Their successful responses can therefore provide complementary supervision (Section 2.2). These observations motivate the approach illustrated in

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.

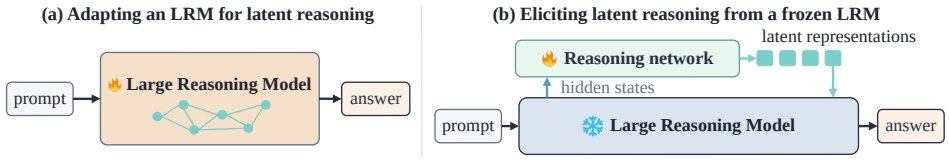


Figure 1: Two routes to latent reasoning. (a) Adapt the LRM itself. (b) Train an external reasoner while keeping the LRM frozen, as in ThinkBridge. Flames and snowflakes mark trainable and frozen components, respectively.

Figure 1(b). We keep the LRM frozen and train an external reasoner to construct latent states, using the model's own successful responses as supervision. Prior work also uses continuous representations to guide frozen language models, supporting this approach (Xu et al., 2025; Jiang et al., 2026; Liu et al., 2025).

Drawing on these observations, we propose ThinkBridge, a framework for eliciting latent reasoning from frozen LRMs. As shown in Figure 3, a lightweight reasoner R constructs a compact sequence of latent states z from the hidden states produced when the frozen LRM encodes the question. The LRM generates its response from the question and this sequence without first generating an explicit CoT.

We aim to train the reasoner to produce question-relevant latent states that take on the guiding role of CoT and support correct responses from the frozen LRM. We train the reasoner with three objectives. First, response supervision trains the reasoner using correct responses generated by the LRM on the seed questions in thinking and non-thinking modes.

Second, Match uses on-policy distillation (Agarwal et al., 2024) to train the reasoner so that its latent states take on the role of CoT in guiding response generation. On response prefixes generated under latent conditioning, we align the frozen LRM's predictions conditioned on latent states with those conditioned on the complete CoT.

Third, Specificity uses contrastive learning to encourage question-specific latent states that support reasoning about the current question. Using full-CoT-conditioned predictions as a common reference, it encourages predictions conditioned on latent states from the current question to be closer to this reference than those conditioned on latent states from other questions.

Our main contributions are as follows.

- We introduce ThinkBridge for frozen LRMs. A lightweight reasoner constructs compact latent states that guide the LRM to respond without first generating explicit CoT. Three objectives train it for response accuracy, CoT functionality, and question specificity. The reasoner's trainable parameters amount to about 5% of the backbone's parameter count.
- Preliminary experiments in mathematical reasoning characterize fine-tuning effects on LRM accuracy and native generation style, and complementary successes of the original LRMs in thinking and non-thinking modes. These findings motivate a frozen backbone and native-response supervision.
- We evaluate ThinkBridge with frozen Qwen3-0.6B and Qwen3-4B on five mathematical reasoning benchmarks and a multi-turn task, assessing accuracy, response latency, and generalization to multi-turn interaction. ThinkBridge achieves the highest mean accuracy among the evaluated latent reasoning methods in both single-turn and multi-turn evaluation.

2 Analysis of LRM Reasoning Behavior

2.1 Effects of Fine-Tuning

We conduct preliminary experiments in mathematical reasoning to examine how existing training methods affect LRMs, comparing changes in accuracy and generation style across models. We study GPT-2, Qwen2.5-0.5B-Instruct, Qwen3-0.6B, and Qwen3-4B, using each unmodified model as a reference and additionally evaluating the two Qwen3 models in thinking mode. We train models

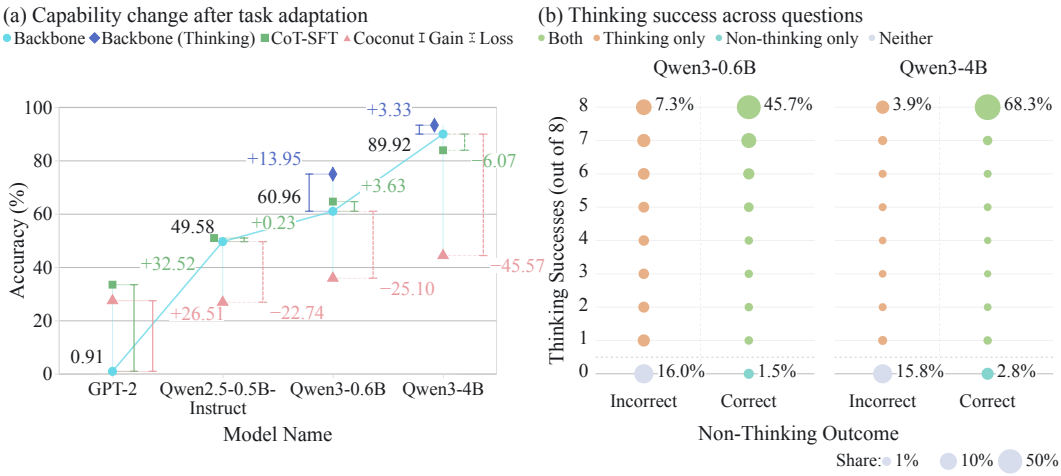


Figure 2: Preliminary observations. (a) GSM8K test accuracy; brackets show changes relative to Backbone. (b) Success patterns of thinking and non-thinking modes; bubble sizes indicate relative question counts.

with CoT-SFT and the latent reasoning method Coconut on GSM8K-Aug-NL and evaluate them on the GSM8K test set. CoT-SFT uses the dataset's original CoTs for supervision. In Figure 2(a), Backbone denotes ordinary answering for GPT-2 and Qwen2.5 and non-thinking mode for Qwen3; Backbone (Thinking) denotes thinking mode.

Figure 2(a) shows that the gains from these training methods differ across models. Both CoT-SFT and Coconut improve accuracy on GPT-2. On Qwen2.5-0.5B-Instruct, CoT-SFT remains close to the original model, while Coconut reduces accuracy. For Qwen3, Coconut falls below the original non-thinking accuracy at both sizes, as does CoT-SFT at 4B. CoT-SFT improves over non-thinking at 0.6B, but neither method reaches thinking accuracy at either size. Thus, the gains observed on GPT-2 do not consistently extend to LRMs, and training can instead reduce their original problem-solving accuracy.

CoT-SFT also produces markedly shorter text than the original model in thinking mode for both Qwen3 models, reflecting a change in generation style. Details of the length and composition measurements, together with Figure B1, are given in Appendix B.

2.2 Complementarity of Thinking and Non-Thinking Modes

Meanwhile, LRMs commonly offer different levels of reasoning effort or thinking modes. We design a preliminary experiment to examine their problem-solving performance under different settings. We use difficulty-stratified sampling to select a seed subset of 14,475 questions from the training split of GSM8K-Aug-NL (see Section 3.1 for data preparation). We then collect responses from the unmodified Qwen3-0.6B and Qwen3-4B in both thinking and non-thinking modes. This subset also provides the questions used to construct ThinkBridge's training data. For each model and question, we collect eight thinking rollouts and one non-thinking response, and summarize their success patterns by final-answer correctness. Sampling and generation settings are given in Appendix A.3.

In Figure 2(b), thinking mode solves some questions that non-thinking mode misses, while some non-thinking successes are missed by all eight recorded thinking attempts. These results show that successful responses from the two modes can provide complementary supervision.

3 Method

ThinkBridge is a latent reasoning framework for a frozen LRM F. Its training pipeline consists of data preparation and reasoner training, as illustrated in Figure 3. We first collect successful responses from F in thinking and non-thinking modes, retaining the paired CoTs for successful

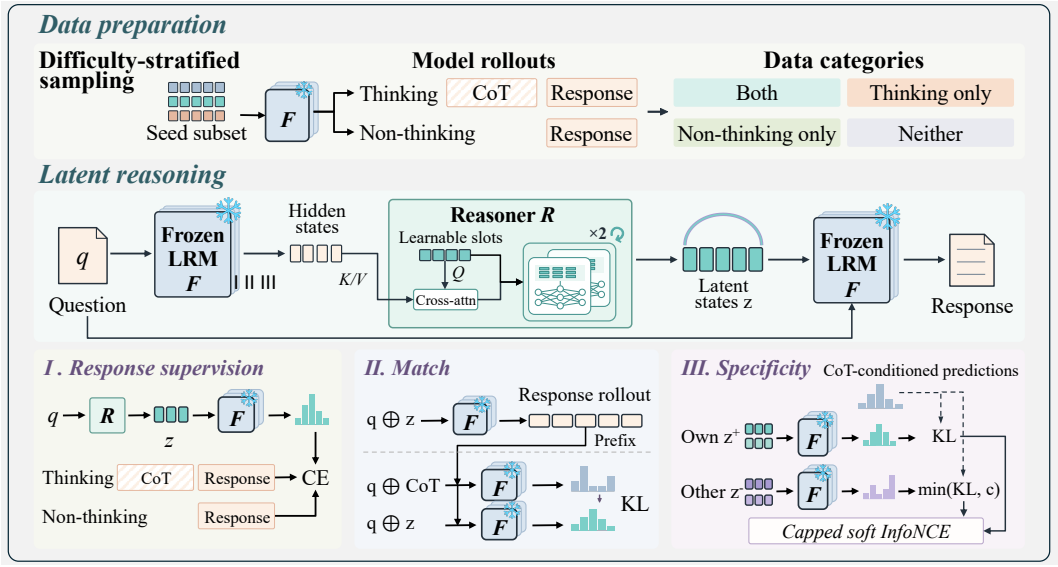


Figure 3: ThinkBridge overview. The reasoner R constructs latent states for response generation by the frozen LRM F. All occurrences of F denote the same model. Data preparation and training objectives are detailed in Sections 3.1 and 3.3. Miniature distributions are schematic.

thinking trajectories. These texts supply native response targets and complete CoTs for training the lightweight reasoner R_θ. This core module constructs compact latent states from the hidden states produced by F for the question, guiding F to respond without first generating explicit CoT. We train R_θ with response supervision, Match, and Specificity, while F remains frozen throughout.

3.1 Data Preparation

We sample a seed subset from the training split of GSM8K-Aug-NL, stratifying by difficulty and arithmetic operation types to cover different levels of complexity and forms of computation. Appendix A.3 details the sampling procedure. The frozen LRM generates thinking rollouts (c_{ij}, a_{ij}) and one non-thinking response for each question q_i. Each CoT c_{ij} remains paired with the complete response a_{ij} from the same rollout. For training, thinking success requires at least one complete trajectory with a correct response; non-thinking success is determined by the single recorded response. These outcomes define four data categories in Figure 3: Both ($\mathcal{D}_{\text{both}}$), Thinking only ($\mathcal{D}_{\text{think}}$), Non-thinking only ($\mathcal{D}_{\text{nonthink}}$), and Neither ($\mathcal{D}_{\text{neither}}$).

The reasoner learns from complete thinking trajectories with correct responses in $\mathcal{D}_{\text{both}} \cup \mathcal{D}_{\text{think}}$ and correct non-thinking responses in $\mathcal{D}_{\text{nonthink}}$. $\mathcal{D}_{\text{neither}}$ provides no successful target and is excluded. Each eligible trajectory or non-thinking response forms one training record. Appendix A.2 describes record sampling and loss averaging.

3.2 Latent Reasoning

Let q denote the question and any conversational context. The frozen LRM F encodes this input once, yielding final-layer hidden states $C(q)$ for all valid question tokens. The reasoner R_θ maps these states to a compact sequence of latent states $z \in \mathbb{R}^{N \times d_F}$ that conditions response generation, where N is the number of latent slots and d_F is the LRM's hidden width.

To map variable-length question states into a fixed number of slots, we use an input cross-attention layer (Cross-attn). The states of N learnable slots, shared across questions, supply its queries Q, while $C(q)$ supplies the keys K and values V. Adding the attention output to the query states initializes the slots with information about the current question. This layer reads the question states once at initialization; subsequent computation operates on the slots.

In our implementation, a looped Transformer (Yang et al., 2024; Zhu et al., 2025) updates the initialized slots to construct z. Appendix A.1 details the network configuration and final amplitude control.

The frozen LRM receives the original question followed by z and generates a response without first generating explicit CoT:

$$z = R_\theta(C(q)), \qquad F(q, z) \to a. \tag{1}$$

Response a includes the final answer used for scoring and any explanation. Constructing z requires only the question; CoT provides supervision during training. Training losses backpropagate through the frozen LRM to R_θ, while the LRM's parameters, token embeddings, and output head remain frozen.

3.3 Reasoner Training Objectives

The three objectives in Figure 3 train the reasoner for response accuracy, CoT functionality, and question specificity. Response supervision uses the LRM's correct responses as targets. Match aligns response predictions under latent and complete-CoT conditioning, and Specificity compares predictions conditioned on current-question and other-question latent states against the same CoT-conditioned reference. The combined objective is

$$L_R = L_{\mathrm{CE}} + L_{\mathrm{match}} + L_{\mathrm{specific}}. \tag{2}$$

We use ℓ_i for per-record losses and L for aggregated objectives. Output dropout produces P latent views $z_i^{(p)}$ for each training record i. This use of dropout to construct views follows SimCSE (Gao et al., 2021). Below, $\mathcal{D}_{\mathrm{both}}$, $\mathcal{D}_{\mathrm{think}}$, and $\mathcal{D}_{\mathrm{nonthink}}$ denote the sets of training records in the corresponding data categories within one parameter update.

Response supervision. For $\mathcal{D}_{\mathrm{both}} \cup \mathcal{D}_{\mathrm{think}}$, we use the responses paired with successful thinking trajectories; for $\mathcal{D}_{\mathrm{nonthink}}$, we use correct non-thinking responses. Cross-entropy supervision on these native responses trains the reasoner to construct latent states that support correct responses from the frozen LRM. Let a_i include the EOS token, and let b denote the response boundary. Then

$$\ell_i^{\mathrm{CE}} = -\frac{1}{P|a_i|}\sum_{p=1}^{P}\sum_{t}\log p_F(a_{it} \mid q_i, z_i^{(p)}, b, a_{i,<t}), \quad L_{\mathrm{CE}} = \mathcal{B}_{\mathcal{D}_{\mathrm{think}},\mathcal{D}_{\mathrm{both}}\cup\mathcal{D}_{\mathrm{nonthink}}}[\ell^{\mathrm{CE}}]. \tag{3}$$

Here $\mathcal{B}$ equally averages the nonempty group means, balancing thinking-only successes with questions already solved in non-thinking mode. Its definition, including empty groups, is given in Appendix A.2.

Match. For $\mathcal{D}_{\mathrm{both}} \cup \mathcal{D}_{\mathrm{think}}$, the paired complete CoT supplies teacher predictions. Through on-policy distillation (Agarwal et al., 2024), we train the reasoner to produce latent states that take the place of CoT in guiding response generation. At the Response rollout step in Figure 3, the frozen LRM greedily generates a response $\hat{a}_i$ conditioned on the first latent view. The same LRM then acts as teacher and student, conditioned on the complete CoT and latent states, respectively. Both predict the next token from the same generated response prefixes:

$$T_{it} = p_F(\cdot \mid q_i, c_i, b, \hat{a}_{i,<t}), \quad S_{it}^{+(p)} = p_F(\cdot \mid q_i, z_i^{(p)}, b, \hat{a}_{i,<t}). \tag{4}$$

We align their predictions using the forward KL from teacher to student over the full vocabulary, averaged over response tokens:

$$d_{i,p}^{+} = \frac{1}{|\hat{a}_i|}\sum_{t} D_{\mathrm{KL}}\Big(\mathrm{stopgrad}(T_{it}) \| S_{it}^{+(p)}\Big), \quad L_{\mathrm{match}} = \frac{\sum_{i\in\mathcal{D}_{\mathrm{both}}\cup\mathcal{D}_{\mathrm{think}}}\sum_{p=1}^{P} d_{i,p}^{+}}{P|\mathcal{D}_{\mathrm{both}}\cup\mathcal{D}_{\mathrm{think}}|}. \tag{5}$$

Teacher probabilities and generated tokens are detached.

Specificity. For each record i in $\mathcal{D}_{\mathrm{think}}$, we compare response predictions conditioned on current-question and other-question latent states. In Figure 3, Own z^+ comprises P_i dropout views from the current record and other same-prompt records in the update, serving as positives. Other z^- supplies

one negative view per selected record $j \in \mathcal{J}_i$, drawn from other questions in $\mathcal{D}_{\text{both}} \cup \mathcal{D}_{\text{think}}$ within that update.

Holding the current question q_i, generated response prefix, and CoT-conditioned teacher predictions fixed, we compute positive distances $d^+_{i,p}$ and negative distances d^-_{ij} using the forward KL in Equation 5. We average positive scores and similarity-weighted negative scores separately:

$$S_i = \frac{1}{P_i}\sum_{p=1}^{P_i}\exp(-d^+_{i,p}/\tau), \quad N_i = \frac{1}{|\mathcal{J}_i|}\sum_{j\in\mathcal{J}_i}\text{stopgrad}(w_{ij})\exp\big(-\min(d^-_{ij}, c)/\tau\big)\,,$$
$$\ell_i^{\text{specific}} = \log\left(1 + \frac{N_i}{S_i}\right), \qquad L_{\text{specific}} = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}\ell_i^{\text{specific}}. \tag{6}$$

Here w_{ij} are detached weights based on prompt-embedding similarity, τ is the contrastive temperature, c is the negative-KL cap, and $\mathcal{I} = \{i \in \mathcal{D}_{\text{think}} : |\mathcal{J}_i| > 0\}$. This multi-positive objective, labeled Capped soft InfoNCE in Figure 3, encourages predictions conditioned on Own latent states to match the same CoT teacher predictions more closely than those conditioned on Other latent states. Capping the negative KL limits the incentive to move the latter predictions farther from the teacher. Appendix A.2 details sampling, weighting, and gradient handling.

4 Experiments

4.1 Experimental Setup

Backbones and baselines. We use Qwen3-0.6B and Qwen3-4B (Yang et al., 2025) as backbones. Baselines include their unmodified thinking and non-thinking modes, No-CoT-SFT, CoT-SFT, Coconut (Hao et al., 2025), CODI (Shen et al., 2025), and PCCoT (Wu et al., 2025). The selected latent reasoning baselines omit the explicit thinking phase and generate only the response. The trained baselines adapt the backbone through full-parameter fine-tuning or LoRA, as indicated in the tables. For the latent baselines, we retain each method's latent construction and training schedule. ThinkBridge keeps the backbone frozen and trains the external reasoner.

Datasets. For ThinkBridge, each backbone generates its own supervision on 14,475 questions sampled from the GSM8K-Aug-NL training split (Deng et al., 2023), as described in Section 3.1. Single-turn evaluation uses GSM8K (Cobbe et al., 2021), GSM-Hard (Gao et al., 2023), MATH-500 (Lightman et al., 2024), MultiArith (Roy & Roth, 2015), and SVAMP (Patel et al., 2021). We use MathChat's three-turn follow-up task (Liang et al., 2024) to evaluate multi-turn performance. All trained methods use GSM8K-Aug-NL's separate 500-question validation split for checkpoint selection. Test-set sizes and dataset-specific protocols are given in Appendix A.5.

Metrics. We report final-answer accuracy, scored using the shared boxed-answer judge, and time to first response token (TTFT). TTFT measures latency from input submission until users receive the first response token, including any preceding thinking. Responses may include explanations. Decoding settings and timing protocols for each method are detailed in Appendix A.5.

Implementation details. R uses a two-layer Transformer stack applied twice and 64 latent slots. We train R for two epochs with native supervision. Architecture, training settings, and checkpoint selection are detailed in Appendices A.1, A.3, and A.4, respectively.

4.2 Single-Turn Reasoning

We first evaluate whether ThinkBridge supports accurate responses with low response latency across mathematical benchmarks. In Table 1, it achieves mean accuracies of 63.53% and 84.88% at 0.6B and 4B, respectively, the highest among the evaluated latent reasoning methods. It achieves the highest accuracy in this group on four of five benchmarks at 0.6B and on all five at 4B, including a tie on MultiArith, so the gains extend beyond GSM8K.

Relative to the original model, ThinkBridge approaches thinking accuracy at 4B, falling 0.58 percentage points short while exceeding non-thinking by 1.09 points. At 0.6B, it is 0.44 points below non-thinking and further from thinking. We also compare the sets of questions solved by

Table 1: **Accuracy (%) and mean TTFT (ms) on five math benchmarks.** Avg is the unweighted benchmark mean; F/L denote full-parameter/LoRA fine-tuning. **Best** and second-best accuracies (including ties) are marked only among latent methods within each model size and column.

Method	GSM		MATH-500	MultiArith	SVAMP	Avg	
	8K	Hard				Acc. ↑	TTFT ↓
Backbone: Qwen3-0.6B							
Non-thinking	60.96	31.39	50.40	94.44	82.67	63.97	20.74
Thinking	74.91	40.11	65.20	96.67	86.67	72.71	25,053.82
No-CoT-SFT^{F}	36.09	8.49	7.20	78.33	48.67	35.76	19.72
No-CoT-SFT^{L}	29.57	7.20	8.60	72.22	50.67	33.65	19.92
CoT-SFT^{F}	64.59	29.57	18.20	98.89	73.00	56.85	1,436.46
CoT-SFT^{L}	60.58	29.87	22.20	97.78	71.00	56.29	1,393.82
Coconut^{F}	35.86	8.26	4.20	67.22	47.33	32.57	218.66
Coconut^{L}	24.11	5.08	10.60	55.56	41.33	27.34	445.76
CODI^{L}	47.54	11.52	3.40	**95.56**	60.00	43.60	334.70
PCCoT^{L}	33.13	7.66	1.80	77.22	51.67	34.30	296.34
ThinkBridge	**63.84**	**32.07**	**50.40**	95.00	**76.33**	**63.53**	45.50
Backbone: Qwen3-4B							
Non-thinking	89.92	55.57	82.00	99.44	92.00	83.79	26.40
Thinking	93.25	59.14	82.60	98.33	94.00	85.46	43,459.22
No-CoT-SFT^{L}	48.60	18.20	23.20	96.11	75.67	52.36	26.92
CoT-SFT^{L}	83.85	53.15	48.40	99.44	82.00	73.37	1,988.50
Coconut^{L}	44.35	14.40	25.80	97.22	67.33	49.82	606.00
CODI^{L}	57.01	15.62	16.00	**99.44**	75.00	52.61	469.84
PCCoT^{L}	50.42	15.01	19.20	95.00	70.00	49.93	403.36
ThinkBridge	**91.58**	**58.30**	**82.40**	**99.44**	**92.67**	**84.88**	56.33

ThinkBridge and the backbone. ThinkBridge answers 74.3% and 93.3% of the questions solved by thinking at 0.6B and 4B, respectively, and also answers some questions that thinking misses. Appendix F gives the full breakdown of these outcomes.

Mean TTFT, including latent construction, is 45.50 ms and 56.33 ms at 0.6B and 4B, respectively, and varies little across benchmarks. Per-benchmark timings are reported in Table G1. A single-turn example is provided in Appendix E.1.

4.3 Ablation Studies

We examine the effects of training objectives and supervision data using the same reasoner architecture with 64 latent slots. Slot-count results are given in Appendix C. To assess how the objectives work together, we remove Match, Specificity, or both while retaining CE and two dropout views. All settings use two epochs of native supervision (Appendix A.3). Table 2 shows that the full objective achieves the highest mean accuracy of 63.53%, compared with 62.52% for CE alone. Removing Match reduces the mean to 63.17%, while removing Specificity gives the largest drop, to 62.02%. CE supervises correct responses. Match directly aligns latent-conditioned and CoT-conditioned predictions on Both and Thinking-only examples. On Thinking-only examples, Specificity compares predictions conditioned on current- and other-question latent states; its positive term also aligns current-question predictions with the same CoT teacher. Removing Match retains this alignment, so the ablation tests the additional direct alignment it provides across both groups.

To assess native data preparation, we replace native supervision with the dataset's original CoTs and answers on the same seed questions, retaining all three objectives. This variant averages 59.17%

Table 2: **Accuracy (%) ↑ with different training objectives.** All runs use Qwen3-0.6B with 64 latent slots. Avg is defined in Table 1.

Variant	GSM8K	GSM-Hard	MATH-500	MultiArith	SVAMP	Avg
ThinkBridge	63.84	32.07	50.40	95.00	76.33	63.53
CE only	62.40	31.39	49.80	91.67	77.33	62.52
w/o Match	63.53	32.22	52.20	92.22	75.67	63.17
w/o Specificity	60.80	31.54	49.00	96.11	72.67	62.02

Table 3: **Accuracy (%) and TTFT (ms) on MathChat's three-turn follow-up subset.** Avg pools all three turns. F/L denote full-parameter/LoRA fine-tuning. **Best** and second-best accuracies are marked per model size and column among latent reasoning methods, including ties.

Method	Turn 1		Turn 2		Turn 3		Avg	
	Acc.↑	TTFT ↓	Acc.↑	TTFT ↓	Acc.↑	TTFT ↓	Acc.↑	TTFT ↓
Backbone: Qwen3-0.6B								
Non-thinking	62.77	20.9	25.84	23.5	16.51	30.8	35.04	25.1
Thinking	75.38	13,126.1	32.03	6,578.1	22.71	7,316.7	43.37	9,007.0
No-CoT-SFT^{F}	36.54	20.6	10.47	21.2	5.12	21.8	17.38	21.2
No-CoT-SFT^{L}	29.28	20.5	10.17	21.2	5.73	21.9	15.06	21.2
CoT-SFT^{F}	63.76	1,405.3	28.21	1,362.0	18.04	1,451.6	36.67	1,406.3
CoT-SFT^{L}	60.93	1,427.5	16.21	1,371.1	11.09	1,454.2	29.41	1,417.6
Coconut^{F}	35.86	220.7	6.96	226.7	3.67	231.5	15.49	226.3
Coconut^{L}	24.16	465.7	8.41	470.9	6.50	481.6	13.02	472.7
CODI^{L}	47.94	329.6	5.28	331.8	1.76	333.1	18.32	331.5
PCCoT^{L}	37.39	323.1	5.96	323.8	3.21	325.1	15.52	324.0
ThinkBridge	**63.69**	45.9	**28.13**	47.7	**15.98**	50.5	**35.93**	48.0
Backbone: Qwen3-4B								
Non-thinking	91.97	28.4	70.80	41.1	62.77	70.8	75.18	46.8
Thinking	93.65	26,738.6	71.79	14,698.0	65.83	16,906.7	77.09	19,447.8
No-CoT-SFT^{L}	48.47	27.5	24.16	29.3	17.13	31.7	29.92	29.5
CoT-SFT^{L}	84.02	1,907.8	58.49	1,744.7	47.09	1,841.3	63.20	1,831.3
Coconut^{L}	44.34	596.5	20.49	607.6	13.61	621.4	26.15	608.5
CODI^{L}	56.88	482.6	11.47	485.8	7.19	486.3	25.18	484.9
PCCoT^{L}	47.86	404.6	11.77	405.7	9.33	405.3	22.99	405.2
ThinkBridge	**91.90**	58.1	**69.34**	95.4	**61.77**	157.8	**74.34**	103.8

and 80.64% at 0.6B and 4B (Table D1), above the latent baselines in Table 1. Native supervision improves all five benchmarks, with mean gains of 4.36 and 4.24 percentage points, respectively. Appendix D details the two training configurations.

4.4 Generalization to Multi-Turn Reasoning

To test generalization beyond single-turn training, we evaluate the same reasoner on 1,308 three-turn mathematical conversations without further training. At each turn, R rebuilds latent states from the current question and all previous questions and generated responses; F receives the same context. Table 3 reports per-turn accuracy and TTFT, with protocols in Appendix A.5.

ThinkBridge achieves the highest accuracy among the evaluated latent reasoning methods at every turn for both model sizes, with mean accuracies of 35.93% and 74.34% at 0.6B and 4B, respec-

tively. At 4B, turn-two and turn-three accuracies of 69.34% and 61.77% remain close to 70.80% and 62.77% in non-thinking mode. These results show that the reasoner also supports follow-up questions that depend on dialogue history.

At 4B, TTFT rises from 58.1 ms to 95.4 ms and 157.8 ms as full responses accumulate in later inputs, increasing history encoding and prefill costs. ThinkBridge uses native-response supervision, whereas trained baselines use dataset-provided targets. Section 2.1 shows that fine-tuning on the dataset's CoTs produces shorter outputs than the original model in thinking mode. Multi-turn TTFT therefore reflects both reasoning computation and history length shaped by generation style.

Appendix E.2 gives a dialogue with correct responses at all three turns.

5 Related Work

From textual to latent reasoning. To reduce the cost of generating explicit CoT token by token, prior work explores intermediate reasoning in continuous representations. Implicit CoT distills explicit CoT into hidden computation, while iCoT gradually removes explicit reasoning steps during training (Deng et al., 2023; 2024). Coconut recursively reuses final hidden states as input embeddings (Hao et al., 2025). CODI jointly learns explicit and implicit reasoning through hidden-state self-distillation (Shen et al., 2025), and PCCoT refines continuous thought tokens through Jacobi iteration (Wu et al., 2025). SIM-CoT uses textual step supervision to align latent reasoning states with their corresponding reasoning steps (Wei et al., 2026). These methods adapt the response-generating model, whereas ThinkBridge trains an external reasoner for a frozen LRM.

Reasoning with frozen models. Prompt and prefix tuning condition frozen language models on learned continuous inputs (Lester et al., 2021; Li & Liang, 2021). SoftCoT projects soft thoughts from a frozen auxiliary model into a frozen target model, which then generates explicit CoT and an answer (Xu et al., 2025). Soft Thinking recursively feeds back probability-weighted token embeddings without training (Zhang et al., 2026). Latent Reasoning Tuning trains a lightweight network to produce continuous representations for a frozen model using complete solutions (Jiang et al., 2026). Latent Recurrent Thoughts iteratively refines latent representations (Chen & Fu, 2026). ThinkBridge learns from the frozen LRM's successful responses in both thinking and non-thinking modes.

Learning from model-generated reasoning. STaR iteratively generates rationales and fine-tunes on successful ones (Zelikman et al., 2022). GKD distills teacher predictions on student-generated sequences (Agarwal et al., 2024). On-Policy Self-Distillation (OPSD) uses the same model as a reference-conditioned teacher and a question-only student, with supervision on the student's own rollouts (Zhao et al., 2026). For contrastive representation learning, SimCSE uses dropout to construct different views of the same sample (Gao et al., 2021). ThinkBridge combines native response supervision, on-policy distillation, and contrastive learning to train an external reasoner while keeping the LRM frozen.

6 Conclusion

ThinkBridge trains a lightweight external reasoner to construct compact latent states while keeping the LRM frozen. Successful responses from thinking and non-thinking modes provide native supervision. Match aligns response predictions under latent and complete-CoT conditioning, while Specificity uses contrastive learning to encourage latent states that support reasoning about the current question. The resulting states guide the LRM to respond without first generating explicit CoT. Across two Qwen3 model sizes, ThinkBridge achieves the highest mean accuracy among the evaluated latent reasoning methods and also supports multi-turn mathematical dialogue without dialogue-specific training.

Limitations. The reasoner is trained on single-turn mathematical questions, so its training does not directly cover references across turns or changes in problem conditions. Dialogue-specific training and generalization beyond three turns remain unexplored. Supervision is filtered by final-answer correctness, which does not ensure that every intermediate CoT step is correct. Evaluation covers two Qwen3 model sizes and mathematical tasks, with a single training seed.

AI USE STATEMENT

We used generative AI tools to assist with method and experiment design, training-objective formulation, coding, literature review, analysis of experimental outputs, manuscript translation, drafting and revision, and figure and table preparation. Section 3.1 describes model-generated CoTs and responses used for training. The authors have reviewed and revised the AI-assisted manuscript and take responsibility for the final text, claims, code, and other artifacts.

ETHICS STATEMENT

This study uses existing mathematical reasoning datasets and pretrained language models, with model-generated responses providing training supervision. It does not involve human-subject experiments or the collection of new personal data.

REPRODUCIBILITY STATEMENT

Section 2 reports the preliminary experiments. Section 3 describes the construction of native supervision, the reasoner, and its training objectives. Section 4.1 specifies datasets, baselines, and evaluation metrics. Appendix A provides network and loss implementation details, training settings, checkpoint selection, and evaluation protocols. Our code is available at `https://anonymous.4open.science/r/ThinkBridge-C0C8`.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Zhaoliang Chen and Jie Fu. Latent recurrent thoughts: Recurrent refinement of proposed latents for reasoning with frozen llms. *arXiv preprint arXiv:2609.01117*, 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.

Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023.

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.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International conference on machine learning*, pp. 10764–10799. PMLR, 2023.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6894–6910, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.552. URL `https://aclanthology.org/2021.emnlp-main.552/`.

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.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24842–24855, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.1274. URL `https://aclanthology.org/2025.findings-acl.1274/`.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3.

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. URL https://openreview.net/forum?id=CbK7lYbmv8.

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

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243/.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/.

Zhenwen Liang, Dian Yu, Wenhao Yu, Wenlin Yao, Zhihan Zhang, Xiangliang Zhang, and Dong Yu. Mathchat: Benchmarking mathematical reasoning and instruction following in multi-turn interactions. *arXiv preprint arXiv:2405.19444*, 2024.

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 *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Luyang Liu, Jonas Pfeiffer, Jiaxing Wu, Jun Xie, and Arthur Szlam. Deliberation in latent space via differentiable cache augmentation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 39261–39274. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/liu25bc.html.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel 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, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168/.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1202. URL https://aclanthology.org/D15-1202/.

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, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

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, 2025. URL https://openreview.net/forum?id=HvoG8SxggZ.

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

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. Sim-cot: Supervised implicit chain-of-thought. In *International Conference on Learning Representations*, volume 2026, pp. 56721–56742, 2026.

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, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.47. URL https://aclanthology.org/2025.emnlp-main.47/.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23336–23351, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1137. URL https://aclanthology.org/2025.acl-long.1137/.

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.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International conference on learning representations*, volume 2024, pp. 42195–42214, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Wang. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *Advances in Neural Information Processing Systems*, 38:168990–169012, 2026.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Jpxfof0EaS.

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 Technical Details

A.1 Looped Latent Construction and Amplitude Control

The reasoner in Section 3.2 applies the same two-layer Transformer stack twice to exchange information among slots and refine their states without adding parameters for each iteration. Let G_0 and G_1 denote these layers. With learnable queries $U \in \mathbb{R}^{N \times d_F}$ and a fixed initialization encoding e_0, we construct the latent states as follows.

$$\begin{aligned} X = \mathrm{RMS}(U + e_0), \qquad \widetilde{C} = \mathrm{RMS}_C(C(q)), \quad h_0 = X + \mathrm{CrossAttn}_\theta(X, \widetilde{C}), \\ h_{\ell+1} = G_{\ell \bmod 2}(h_\ell), \qquad \ell = 0, 1, 2, 3, \quad v = \mathrm{Dropout}(g \odot h_4), \\ z_k = r_E e^s \frac{v_k}{\sqrt{\tau_{\mathrm{bound}}^2 + \|v_k\|_2^2 / d_F}}, \qquad k = 1, \ldots, N. \end{aligned} \tag{7}$$

We initialize U with zero-mean Gaussian noise of standard deviation 0.02. For $m = \lfloor d_F/2 \rfloor$ and $\omega_j = 10000^{-j/m}$, $j = 0, \ldots, m-1$, e_0 repeats the vector $[\cos\omega; \sin\omega]$ across slots, normalized to unit RMS and scaled by the initial RMS of U.

The input cross-attention layer has bias-free Q/K/V/O projections and per-head Q/K RMSNorm. The normalization of $U + e_0$ is non-affine ($\epsilon = 10^{-6}$); question and Q/K normalizations have learned gains. Each block uses pre-normalized, non-causal self-attention with 16 heads, Q/K RMSNorm and no rotary position embeddings, followed by a bias-free SwiGLU of width $\lfloor 8d_F/3 \rfloor$, with residual connections around both sublayers.

Equation 7 applies channel gain, dropout and amplitude control only to the final loop state. Here r_E is the frozen embedding-table RMS, g starts at one, s starts at $\log 2$, and $\tau_{\text{bound}} = 1$, so $\text{RMS}(z_k) < r_E e^s$. Projections use Xavier initialization, with gain $1/\sqrt{3}$ for loop self-attention output and FFN down projections; normalization gains start at one.

Table A1 reports the additional parameters of R, counting shared loop weights once.

Table A1: Parameter counts at $N = 64$. R contains the added trainable parameters; R/F is their percentage relative to frozen F.

Backbone	F (M)	R (M)	R/F (%)
Qwen3-0.6B	596.05	29.44	4.94
Qwen3-4B	4,022.47	183.69	4.57

A.2 Reasoner Objective

Question template and response boundary. Figure A1 shows the thinking and non-thinking prompts and output formats. We apply the Qwen3 chat template to the user message. The assistant prefix ends in `<think>\n`, and q in our equations includes this formatted input. The teacher receives the complete CoT from the paired rollout, while the student receives z. Both are followed by the fixed boundary b, the token sequence for `\n</think>\n\n`, before response generation. The model's tokenizer encodes the template text; the boundary is not learned. The non-thinking reference uses an empty thinking block. For the multi-turn history described in Section 4.4, the mathematical instruction is appended only to the current question before applying the same chat format. Here `\n` denotes a newline.

	Thinking	**Non-thinking**
Prompt	{question} Let's think step by step and output the final answer within \boxed{}.	{question} Output the final answer within \boxed{}.
Template	<think> {CoT} </think> {response}	<think> </think> {response}

Figure A1: Prompts and output formats for thinking and non-thinking generation.

Response targets and dropout views. For response supervision, the input consists of the question prompt, complete sequence of latent states, response boundary, and saved successful response, in that order. Cross-entropy covers response and EOS tokens, with the last boundary position predicting the first response token. Valid sequence positions are consecutive, and padding is excluded from attention keys.

Response supervision and prediction matching follow Equations 3 and 5, respectively. An EOS produced during generation is included in the loss, whereas length-capped responses receive no additional EOS. Evaluation disables dropout and constructs one sequence of latent states per question.

Loss averaging. The three objectives select eligible examples from the same training records, without fixed sampling quotas for individual categories. To express the equal weighting of category means in response supervision, let $\bar{\ell}_U = |U|^{-1} \sum_{i \in U} \ell_i$ and define

$$\mathcal{B}_{U,V}[\ell] = \begin{cases} (\bar{\ell}_U + \bar{\ell}_V)/2, & U, V \neq \varnothing, \\ \bar{\ell}_U, & U \neq \varnothing,\ V = \varnothing, \\ \bar{\ell}_V, & U = \varnothing,\ V \neq \varnothing, \\ 0, & U = V = \varnothing. \end{cases} \tag{8}$$

This operator is used in Equation 3; Match and Specificity are aggregated as in Equations 5 and 6. An objective contributes zero when no record meets its conditions. All denominators count records across all devices and gradient accumulation steps in one parameter update.

Record sampling and own latent states. At each epoch, we sort eligible trajectory records by stable identifiers, shuffle them with the epoch seed, and form successive updates of 128 records. Records are assigned consecutively to training processes and gradient accumulation steps; the incomplete final update is dropped. Multiple records with the same prompt may occur in one update and retain their positions in the shuffled sequence.

Same-prompt records jointly supply the positive set for Specificity. Each record contributes P independently sampled dropout views, indexed jointly by $p = 1, \ldots, P_i$. All views are scored against the current record's paired CoT teacher. CE and Match use only the current record's two views.

Selection and weighting of other latent states. Candidates come from $\mathcal{D}_{\text{both}} \cup \mathcal{D}_{\text{think}}$ within the same update. We exclude records that share the current prompt, original problem, or semantic group. From the remaining candidates, a seeded sampler uniformly selects up to two records without replacement. We use all available records when fewer than two qualify, and omit Specificity for an example with none. Each selected record contributes one dropout view as a contrastive negative.

These exclusions compare formatted prompt-token sequences, original-problem identifiers, and semantic groups, respectively. Original-problem identity uses the problem ID when available and otherwise the source-question ID. Semantic groups use explicit labels when available. Without such labels, training records are grouped by combinations of arithmetic operation types identified in their reference solutions, as described in Appendix A.3; validation records are grouped by original-problem identity. These rules govern the choice of other-question records, while training and validation retain the dataset's original split.

Weights for other latent states depend on frozen token-embedding similarity. Let $\bar{e}_i$ be the mean of frozen embeddings over valid prompt tokens. The detached weights are

$$a_{ij} = \max\left(\frac{1 + \text{clip}(\cos(\bar{e}_i, \bar{e}_j), -1, 1)}{2}, 10^{-3} \right), \quad w_{ij} = \frac{a_{ij}}{|\mathcal{J}_i|^{-1} \sum_{k \in \mathcal{J}_i} a_{ik}}. \tag{9}$$

The weights have mean one over $\mathcal{J}_i$ and give greater relative weight to records with more similar prompt embeddings. We compute weights and losses in FP32, using log-sum-exp for numerical stability.

We set $L_{\text{specific}} = 0$ when $\mathcal{I}$ in Equation 6 is empty. All positive and negative latent branches train R, including additional same-prompt views and negative views from $\mathcal{D}_{\text{both}}$; teacher predictions, generated prefixes, and weights are detached. Once d_{ij}^{-} reaches c, increasing it no longer reduces the contrastive loss through that negative term. Other terms can still update the shared reasoner.

A.3 Training and Data Details

Supervision data. The training subset and validation split described in Section 4.1 do not overlap in question text or formatted prompt tokens. Each model produces eight thinking trajectories per training question (temperature 1.0, cap 8,192 tokens) and one greedy non-thinking response (cap 2,048 tokens).

Difficulty is estimated from computation steps in the reference CoT. We split at whitespace after periods and count fragments containing = or case-insensitive `equals`, with a minimum count of

one. Counts of one, two–three, and at least four define easy, medium, and hard questions. Symbol and word rules identify addition, subtraction, multiplication, division, fractions, and percentages. We stratify by difficulty and matched operation types, uniformly sample 100 records per group without replacement (or retain all records in smaller groups), then merge and shuffle. The 188 groups yield 5,125 easy, 5,370 medium, and 3,980 hard questions.

Thinking trajectories used for training require correct final answers, normal termination, a closed thinking block, and nonempty CoT and response. Table A2 uses at least one such trajectory among eight attempts to define thinking success; non-thinking success uses the single response. Unlike these counts, Figure 2(b) uses correctness alone. At 0.6B, completion filtering moves two questions from $\mathcal{D}_{\mathrm{both}}$ to $\mathcal{D}_{\mathrm{nonthink}}$ and 22 from $\mathcal{D}_{\mathrm{think}}$ to $\mathcal{D}_{\mathrm{neither}}$; the corresponding counts at 4B are 42 and 55.

Table A2: Question counts after training-data filtering. Categories are defined in Section 3.1.

Backbone	$\mathcal{D}_{\mathrm{both}}$	$\mathcal{D}_{\mathrm{think}}$	$\mathcal{D}_{\mathrm{nonthink}}$	$\mathcal{D}_{\mathrm{neither}}$
Qwen3-0.6B	8,448	3,459	226	2,342
Qwen3-4B	10,475	1,209	453	2,338

Optimization. Table A3 gives the reasoner's optimizer settings. All three R objectives are active from the first update, with dropout 0.1, two views per record, and up to two other-question negatives. Specificity uses $c = 0.2$ nats per response token and $\tau = 0.1$; the teacher and student KL distributions use softmax temperature 1.

Table A3: Training settings for the reasoner.

Setting	Value
Optimizer	AdamW
Learning-rate schedule	Cosine
Learning rate	2×10^{-4}
Weight decay	0.01
Gradient clipping	1
Global batch size	128
Training epochs	2
Warmup updates	40
Evaluation/save interval (updates)	50
Trainable parameters and Adam states	FP32
Main matrix operations	BF16 autocast

The reasoner is also evaluated and saved at the final update; checkpoint selection is described in Appendix A.4. Responses used to supply training prefixes are generated greedily, with a 2,048-token limit. The training seed is 42, with shuffle seed $42 + e$ for zero-based epoch e.

Distributed execution. We use a server with eight NVIDIA GPUs, each with 96 GB of memory, for training and inference. Training uses four GPUs for response generation and four for parameter updates. Table A4 specifies batching. Additional positive and negative candidates are scored in chunks of four. We use DDP for 0.6B and ZeRO-1 for 4B, retaining FP32 normalization and amplitude control. Evaluation precision, decoding and batching are given in Appendix A.5.

Baseline training. Table A5 gives baseline settings; CODI uses six latent states, and PCCoT uses 24 with three parallel iterations. Objective ablations use the training configuration above, with variants defined in Section 4.3. Slot-count and supervision-data ablations are described in Appendices C and D, respectively.

Table A4: Training batches and gradient checkpointing. Records/GPU counts records per accumulation step; compute batch is the number processed together.

Model	Records/ GPU	Accum. steps	Compute batch	Gradient checkpointing
0.6B	16	2	16	Off
	8	4	8	Off
4B	8	4	4	Off

Table A5: Baseline training settings. SFT includes No-CoT-SFT and CoT-SFT; LR denotes learning rate, and r/α denotes LoRA rank/scale.

Method	Adaptation	Epochs	LR (0.6B)	LR (4B)	LoRA r/α
SFT	Full	10	10^{-5}	10^{-5}	—
SFT	LoRA	10	10^{-4}	10^{-4}	16/32
CODI/PCCoT	LoRA	10	8×10^{-4}	2×10^{-4}	128/32
Coconut	Full	25	10^{-4}	10^{-4}	—
Coconut	LoRA	25	10^{-4}	10^{-4}	128/32

A.4 Checkpoint Selection

On the validation split specified in Section 4.1, ThinkBridge first collects the required model responses and selects up to 500 questions using data categories and a deterministic hash ordering. We choose the checkpoint with the highest response accuracy on this fixed set, conditioned on the question and latent states. Ties are broken in favor of the later checkpoint.

A.5 Evaluation Protocol

Benchmarks. Single-turn evaluation uses GSM8K (1,319 problems) (Cobbe et al., 2021), GSM-Hard (1,319) (Gao et al., 2023), MATH-500 (500) (Lightman et al., 2024), MultiArith (180) (Roy & Roth, 2015), and SVAMP (300) (Patel et al., 2021). Multi-turn evaluation uses 1,308 three-turn conversations from MathChat (Liang et al., 2024).

Decoding and scoring. ThinkBridge validation and evaluation use HuggingFace with BF16 computation and SDPA attention. We use greedy decoding with a response limit of 2,048 tokens. To reduce variation across runs, we fix the random seed to 42 and disable dropout and TF32 matrix multiplication during evaluation. All methods use the shared boxed-answer judge to score final-answer accuracy.

Multi-turn evaluation. The history in Section 4.4 retains complete responses, including any explanation, with all earlier questions and responses passed verbatim to the next turn. ThinkBridge uses eight independent replicas, each with batch size one. Accuracy and TTFT are computed from the same complete responses. Each turn includes all 1,308 conversations, giving 3,924 responses across three turns. The Avg column in Table 3 includes all three turns.

Time to first response token. Single-turn ThinkBridge timing uses batch size one for both the reasoner and LRM. TTFT starts before input preparation and ends at the first response token, including question encoding, latent construction, and LRM prefill. Thinking and control tokens do not count as response tokens. We warm up the model before timing and exclude model loading and warmup. Generation stops after the first response token; accuracy is measured in separate full-generation runs of the same checkpoint. The backbone's thinking and non-thinking timings cover model computation and exclude tokenization and prompt formatting, whereas ThinkBridge timing also includes these input preparation steps.

In multi-turn timing, the next request arrives immediately after the previous response ends. History construction counts toward that request's TTFT, followed by question encoding, latent construction, and LRM prefill. Both TTFT means and accuracy estimates use the complete sample described above.

B Output Length after Fine-Tuning

Section 2.1 discusses the observed length changes; here we explain the measurements and visual encoding used in Figure B1. C denotes intermediate text (six latent steps for Coconut), A denotes final-answer tokens, and hatching marks text inside <think>. Labels give total lengths and approximate composition percentages. To display outputs of different lengths together, horizontal positions use $\log(1 + \text{length})$, with a minimum visible width for answer segments. The ≥ 64 label marks unstructured GPT-2 output.

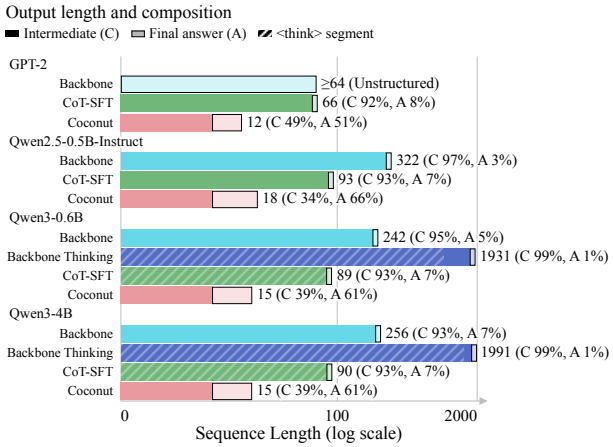


Figure B1: Output length and composition after fine-tuning. Hatching marks text inside <think>.

C Effect of Latent Slot Count

We vary N among 32, 64, and 128, keeping the remaining architecture and training objectives fixed. All settings use two epochs of native supervision and the common training configuration in Appendix A.3. Table C1 shows that mean accuracy is highest with 32 slots at 0.6B and 64 slots at 4B; increasing the slot count does not consistently improve accuracy.

Table C1: **Accuracy (%) ↑ with different numbers of latent slots.** Both models use all three training objectives, with 64 slots as the default. Avg is defined in Table 1.

Backbone	# Slots	GSM8K	GSM-Hard	MATH-500	MultiArith	SVAMP	Avg
Qwen3-0.6B	32	65.05	32.75	49.60	96.67	76.00	64.01
	64	63.84	32.07	50.40	95.00	76.33	63.53
	128	61.03	30.55	46.60	95.00	79.33	62.50
Qwen3-4B	32	91.81	57.77	81.40	99.44	92.00	84.49
	64	91.58	58.30	82.40	99.44	92.67	84.88
	128	91.43	56.86	81.80	100.00	91.33	84.29

D Effect of Native Data Preparation

We compare full ThinkBridge with a variant that omits native data preparation. The variant retains the same 14,475 seed questions but uses their original GSM8K-Aug-NL CoTs and answers instead of generating and filtering model responses. It retains one reference CoT and boxed answer per question. Full ThinkBridge uses eligible model-generated trajectories and responses, as described in Section 3.1; its results are repeated from Table 1. Both settings use the same training objectives, record sampling, and contrastive selection rules. The dataset-supervision condition runs for 1,278 updates at 0.6B and 1,394 at 4B, with a 20-epoch ceiling; native supervision uses the configuration in Appendix A.3. Checkpoint selection follows Appendix A.4.

Table D1: **Effect of native data preparation on accuracy (%) ↑.** Both settings use 64 latent slots and all three training objectives. Avg is defined in Table 1.

Variant	GSM8K	GSM-Hard	MATH-500	MultiArith	SVAMP	Avg
Qwen3-0.6B						
ThinkBridge	63.84	32.07	50.40	95.00	76.33	63.53
w/o native data preparation	55.88	27.29	44.80	92.22	75.67	59.17
Qwen3-4B						
ThinkBridge	91.58	58.30	82.40	99.44	92.67	84.88
w/o native data preparation	86.66	52.62	80.60	96.67	86.67	80.64

Native supervision gives higher accuracy on all five benchmarks at both model sizes (Table D1). Mean accuracy is 63.53% versus 59.17% at 0.6B and 84.88% versus 80.64% at 4B. The two data construction schemes differ in trajectory source, examples per question, and training-record composition. Even with dataset-provided targets, ThinkBridge exceeds the mean accuracy of the evaluated latent baselines in Table 1 at both model sizes. Thus, its advantage in these experiments is not solely due to native supervision.

E CASE STUDIES

E.1 SINGLE-TURN REASONING

Figure E1 shows a correct single-turn response. ThinkBridge combines the initial and additional cookies, then subtracts those eaten to obtain the correct answer of 21.

Question	Rachel bought 23 cookies and Janet gave her 42 cookies. The other day her brother ate 44 of those cookies. How many cookies are left for Rachel?
Response	Rachel initially had 23 cookies. Janet gave her 42 more, so the total number of cookies Rachel had before any were eaten is: 23 + 42 = 65. Then, her brother ate 44 of those cookies. To find out how many are left, we subtract the number of cookies eaten from the total: 65 − 44 = 21. Thus, the number of cookies left for Rachel is \boxed{21}.
Gold answer	21

Figure E1: A correct single-turn response generated by ThinkBridge.

E.2 MULTI-TURN REASONING

Figure E2 shows a dialogue with correct responses at all three turns. The questions progress from purchase cost to resale profit and then reinvestment. The responses use the earlier quantities to compute $4,500 in cost, $600 in profit, and one additional goat, respectively.

Turn 1	**Question**	John goes to the market and buys 3 goats for $500 each and 2 cows for $1500 each. How much money did he spend?
	Response	John buys 3 goats at $500 each and 2 cows at $1500 each. To find the total amount he spent, we calculate the cost of the goats and the cows separately and then add them together. For the goats: 3 × 500 = 1500. For the cows: 2 × 1500 = 3000. Adding these two amounts together: 1500 + 3000 = 4500. Thus, the total amount of money John spent is \boxed{4500}.
	Gold answer	4500
Turn 2	**Question**	If John decides to sell all the goats and cows he bought at a profit of 20% per goat and 10% per cow, how much total profit does he make from the sales?
	Response	John sells all the goats and cows he bought at a profit of 20% per goat and 10% per cow. To determine the total profit, we need to calculate the profit from each type of animal separately and then sum them. John bought 3 goats at $500 each, so the total cost for the goats is 3 × 500 = 1500. He sells each goat at a 20% profit. ... The profit from the goats is 1800 − 1500 = 300. John bought 2 cows at $1500 each, so the total cost for the cows is 2 × 1500 = 3000. He sells each cow at a 10% profit. ... The profit from the cows is 3300 − 3000 = 300. Adding the profits from both goats and cows: 300 + 300 = 600. Thus, John makes a total profit of \boxed{600} dollars from the sales.
	Gold answer	600
Turn 3	**Question**	Suppose John reinvests the total profit from the goats and cows into buying more goats, this time at a discounted price of $450 per goat. How many additional goats can he buy?
	Response	From the previous calculation, we found that John made a total profit of $600 from selling the 3 goats and 2 cows. John reinvests this $600 into buying more goats at a discounted price of $450 per goat. To find how many additional goats John can buy, we divide the total profit by the new price per goat: 600 ÷ 450 = 1.33. This means John can buy 1 full goat with the $600, and he will have $150 left. Since the remaining $150 is not enough to buy another goat at $450, John can buy \boxed{1} additional goat.
	Gold answer	1

Figure E2: A three-turn dialogue with correct ThinkBridge responses at every turn. Ellipses indicate omitted response text.

F Overlap with Backbone Reasoning Outcomes

To complement the mean accuracies in Table 1, Figure F1 compares which questions ThinkBridge and the backbone's thinking and non-thinking modes solve correctly. The proportions weight all questions equally, whereas the main table averages benchmark accuracies.

Among the questions solved by thinking, ThinkBridge solves 74.3% and 93.3% at 0.6B and 4B, respectively, without first generating explicit CoT. It also solves questions that thinking misses, accounting for 6.2% and 4.4% of all questions. Thus, at 4B, the small gap in mean accuracy is accompanied by substantial overlap in the questions solved.

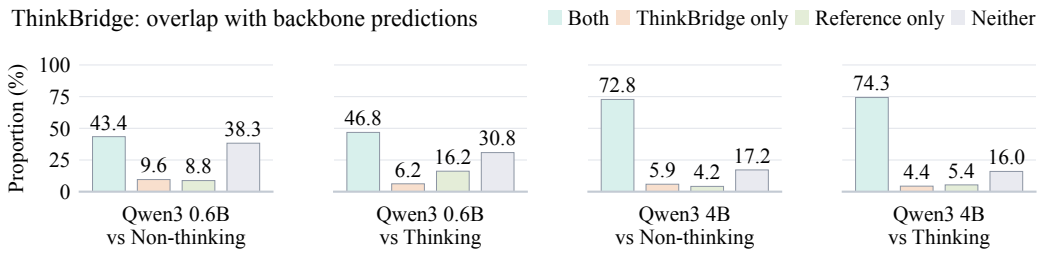


Figure F1: Correctness overlap between ThinkBridge and each backbone mode on the same 3,618 questions pooled across five benchmarks. Each group identifies the model size and reference mode. Both/Neither denote joint success/failure; ThinkBridge only/Reference only denote success by just ThinkBridge/the reference.

G SINGLE-TURN RESPONSE LATENCY

Table G1 provides the per-benchmark timings summarized by the mean TTFTs in Table 1. The timing protocols are described in Appendix A.5.

Table G1: Single-turn TTFT (ms) ↓on each mathematical benchmark. Avg is the unweighted benchmark mean; F/L denote full-parameter/LoRA fine-tuning.

Method	GSM8K	GSM-Hard	MATH-500	MultiArith	SVAMP	Avg
Backbone: Qwen3-0.6B						
Non-thinking	21.2	21.2	21.2	20.0	20.1	20.74
Thinking	12,564.3	57,795.6	40,469.1	6,489.7	7,950.4	25,053.82
No-CoT-SFT^{F}	20.2	20.3	20.1	19.0	19.0	19.72
No-CoT-SFT^{L}	20.5	20.4	20.3	19.2	19.2	19.92
CoT-SFT^{F}	1,430.7	1,960.1	2,161.9	851.5	778.1	1,436.46
CoT-SFT^{L}	1,424.6	1,908.6	2,132.4	768.6	734.9	1,393.82
Coconut^{F}	218.2	217.5	218.5	219.1	220.0	218.66
Coconut^{L}	440.6	448.7	446.6	447.5	445.4	445.76
CODI^{L}	334.1	334.8	334.8	335.1	334.7	334.70
PCCoT^{L}	294.7	296.1	295.3	299.6	296.0	296.34
ThinkBridge	45.60	45.63	45.74	45.28	45.25	45.50
Backbone: Qwen3-4B						
Non-thinking	27.1	27.0	26.8	25.5	25.6	26.40
Thinking	25,717.2	89,148.9	71,134.5	12,615.7	18,679.8	43,459.22
No-CoT-SFT^{L}	27.4	27.5	27.4	26.1	26.2	26.92
CoT-SFT^{L}	2,013.4	2,599.1	3,124.0	1,176.5	1,029.5	1,988.50
Coconut^{L}	604.5	605.0	605.9	607.9	606.7	606.00
CODI^{L}	476.9	475.8	472.5	461.9	462.1	469.84
PCCoT^{L}	402.8	403.5	403.1	405.9	401.5	403.36
ThinkBridge	56.17	56.26	57.68	55.75	55.77	56.33