LaTER: Adaptive Latent Computation for Efficient Explicit Reasoning

Anonymous authors
Paper under double-blind review

Abstract

Chain-of-thought (CoT) reasoning realizes additional inference-time computation through autoregressive token generation, while continuous latent reasoning removes this dependence on discrete intermediate tokens. However, making the entire derivation latent can degrade reasoning accuracy. We propose **La**tent-**T**hen-**E**xplicit **R**easoning (**LaTER**), which uses a continuous latent prefix to prepare a shorter explicit derivation. LaTER recurrently feeds projected hidden states back as inputs, then transitions to token generation while retaining the latent key–value cache. We study both a training-free variant, which determines this transition from vocabulary-probe entropy and terminating-token predictions, and a supervised variant trained on LATENT-SWITCH-69K, a corpus of 69,745 examples that we construct by pairing solution intuitions with shortened derivations. To learn reliable stopping behavior, we introduce a halting objective that accounts for a training–inference mismatch in which forced training rollouts can mask premature stop predictions. On Qwen3-14B, supervised LaTER improves accuracy while reducing reasoning length on all seven evaluated benchmarks relative to both standard CoT and a same-data CoT-SFT control. On AIME 2025, accuracy increases from 70.0% to 80.0%, while latent steps plus explicit tokens decrease from 15,730 to 10,575. On Qwen3-8B, wall-clock decoding time decreases by 19.1% on average across benchmarks in our implementation. These results suggest that latent and explicit reasoning are complementary, and that the transition between them is a key design choice for improving the accuracy–efficiency tradeoff. Our code, data, and model are available at `https://anonymous.4open.science/r/LaTER-EF4D`.

1 Introduction

CoT prompting improves language-model reasoning by externalizing intermediate computation as a sequence of tokens (Wei et al., 2022), with strong performance on mathematics, science, and code tasks (Guo et al., 2025). This capability can be expensive. Under standard autoregressive decoding, generating each additional reasoning token requires another sequential decoding step. Long reasoning traces therefore increase inference latency and memory traffic, while also extending the context over which subsequent tokens attend. In our experiments, the Qwen3-14B CoT baseline generates 15,730 tokens per AIME 2025 problem on average. Efficient reasoning therefore requires deciding not only *how much* computation to allocate, but also *which intermediate computations need to be expressed as text*.

Continuous latent reasoning offers a way to relax this coupling between computation and discrete token generation (Hao et al., 2025; Zhang et al., 2025). Rather than feeding back a sampled token embedding, the model reuses a hidden state or continuous embedding as its next input, allowing intermediate computation to proceed without producing a visible token at every step (Xu et al., 2025; Zou et al., 2026). Latent reasoning does not eliminate computation—each latent step still requires a forward pass—but it can reduce the amount of explicit reasoning that must be generated autoregressively. However, aggressively replacing CoT with latent computation can reduce accuracy on difficult benchmarks such as AIME (Deng et al., 2025; Tan et al., 2025; Shen et al., 2025). Our Qwen3-8B comparisons show the same pattern: several latent methods produce substantially shorter outputs than CoT while achieving lower accuracy across the benchmarks. The objective should therefore be a better accuracy–cost tradeoff, rather than maximal compression of visible reasoning.

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.

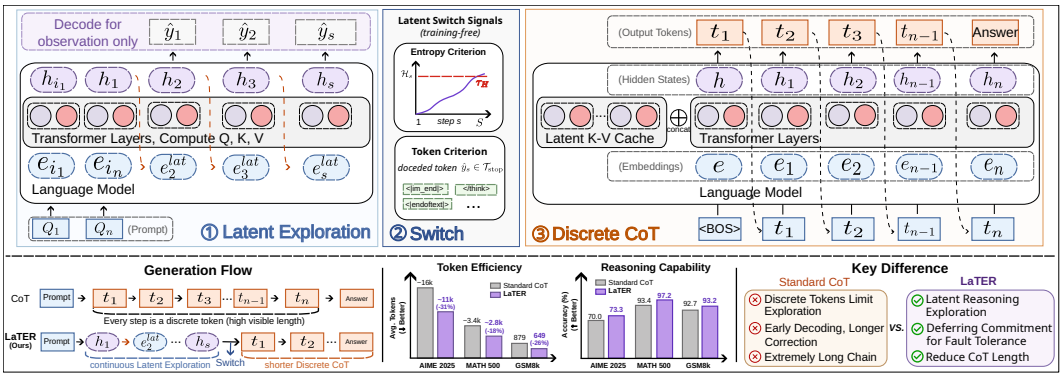

Figure 1: **The LaTER inference interface.** Top: the model first recurrently maps final-layer hidden states back to input embeddings without emitting visible tokens. It then switches to ordinary token generation and retains the latent KV cache when producing the explicit derivation and final answer. Bottom: generation flow and Qwen3-14B results comparing LaTER-TF with CoT.

We study *latent computation as preparation for an explicit derivation*. We propose **LaTER** (**La**tent-**T**hen-**E**xplicit **R**easoning), which performs a recurrent latent rollout before switching to ordinary token generation (Figure 1). Each latent step maps the model's hidden state back into the input embedding space without emitting a visible token. The explicit continuation retains the latent-phase key–value (KV) cache, allowing it to build on the latent prefix. LaTER thus uses latent computation to prepare a shorter explicit derivation.

The central question is *when to switch*. A latent prefix that is too short may add computation without sufficiently shortening the explicit continuation, whereas one that is too long may impair the accuracy of the subsequent explicit derivation. Fixed-horizon latent rollouts shorten generation on AIME 2025 but do not recover standard CoT accuracy. We therefore study two adaptive variants of the same interface. Training-free LaTER (LaTER-TF) uses vocabulary-probe entropy and terminating-token predictions to select the latent-to-explicit transition. Supervised LaTER (LaTER-SFT) learns a projector and stopping behavior from LATENT-SWITCH-69K, a corpus of 69,745 examples that we construct by pairing solution intuitions with shortened explicit derivations.

Learning this transition requires aligning training and inference. Training executes a prescribed latent rollout even if the model predicts a stopping token early, whereas inference exits at that prediction. We therefore use a halting objective to penalize premature structural-token predictions and supervise the intended boundary, making stopping behavior an explicit part of training.

Experiments show that LaTER improves the accuracy–efficiency tradeoff both with and without additional training. On Qwen3-14B, training-free LaTER improves AIME 2025 accuracy from 70.0% to 73.3% while reducing latent steps plus explicit tokens from 15,730 to 10,661. With supervision, LaTER reaches 80.0% accuracy while further reducing this count to 10,575, and it improves over both standard CoT and a same-data CoT-SFT control across all seven evaluated benchmarks. On Qwen3-8B, the resulting shorter reasoning trajectories reduce measured wall-clock decoding time by 19.1% on average across benchmarks in our implementation.

Our contributions are threefold. (i) We introduce a latent-then-explicit reasoning interface that uses continuous computation to prepare, rather than replace, an explicit derivation while preserving the latent KV cache. (ii) We develop adaptive mechanisms for locating the latent-to-explicit boundary, including training-free entropy and terminating-token probes and a halting objective for learned stopping. (iii) We construct LATENT-SWITCH-69K and show that supervised LaTER improves the accuracy–efficiency tradeoff across mathematics, coding, and knowledge-intensive reasoning benchmarks. These results suggest that latent and explicit reasoning are complementary, and that the transition between them is a central design choice for efficient and accurate reasoning.

2 LATENT-THEN-EXPLICIT REASONING

LaTER uses continuous computation as a prefix to an explicit derivation. Given a prompt Q, the model performs S recurrent latent transitions and then generates an explicit reasoning sequence $t_{1:n}$

followed by an answer a. The two phases share one causal history, the explicit continuation retains the KV cache produced during the latent prefix rather than restarting from the prompt.

2.1 A CONTINUOUS PREFIX WITH A DISCRETE CONTINUATION

Let h_s be the final-layer hidden state at step s, and e_{s+1}^{lat} be the embedding at the next position. In the training-free setting, we follow the latent transition used by LatentMAS (Zou et al., 2026):

$$e_{s+1}^{\text{lat}} = W_a h_s, \qquad W_a \approx W_{out}^{\dagger} W_{in}, \tag{1}$$

where W_{in} is the input embedding matrix, W_{out} is the vocabulary projection, and $W_{out}^{\dagger}$ is its pseudo-inverse. The resulting recurrence

$$h_1 \to e_2^{\text{lat}} \to h_2 \to \cdots \to e_S^{\text{lat}} \to h_S \tag{2}$$

adds positions to the causal cache without committing to a discrete intermediate token. Section 3 replaces W_a with a learned projector, but preserves the same latent-prefix interface and the one-way transition to explicit decoding.

To observe the trajectory in vocabulary space, we form a probe distribution, its argmax token, and its entropy:

$$p_s = \text{softmax}(W_{out} h_s), \qquad \hat{y}_s = \arg\max_i p_s(i), \qquad \mathcal{H}_s = -\sum_i p_s(i) \log p_s(i). \tag{3}$$

The probe observes vocabulary alignment and the next input remains the projected continuous state in Equation 1, not the probe token $\hat{y}_s$.

2.2 LOCATING THE TRANSITION

Vocabulary probes of latent rollouts reveal model-native terminating tokens even without discrete token feedback or additional supervision. Probe entropy also changes over the trajectory, with a rising average trend over latent progress (Appendix C). These observations motivate monitoring both stopping-token predictions and entropy during the latent rollout to decide when to return to text. We use them as heuristic signals that allow the transition boundary to vary across samples.

We compare a fixed horizon with an adaptive policy that exits when probe entropy exceeds a threshold or the probe argmax reaches a terminating token:

$$\text{switch}(s) = \mathbb{I}[\mathcal{H}_s > \tau_{\mathcal{H}} \ \vee \ \hat{y}_s \in \mathcal{T}_{\text{stop}}]\,. \tag{4}$$

The model-dependent set $\mathcal{T}_{\text{stop}}$ includes tokens such as `</think>`, `<|im_end|>`, and `<|endoftext|>`. Section 5 evaluates the resulting boundary sensitivity and failure modes.

3 LEARNING THE LATENT PREFIX AND ITS BOUNDARY

Probe-based switching provides a training-free transition rule, but entropy scales vary across samples, and low-entropy prefixes can precede incorrect answers (Appendix C). This motivates jointly learning a latent prefix that supports a shorter explicit derivation and the boundary at which to return to text. We supervise LaTER-SFT with shortened explicit continuations and prescribed latent boundaries derived from solution intuitions.

3.1 LEARNED RECURRENCE AND FORMAT

For Qwen3-8B and Qwen3-14B, we replace the analytic mapping with a two-layer MLP projector g_ϕ with GELU activation. It maps final-layer hidden states into the input embedding space, preserving dimensionality (4,096 and 5,120, respectively). We add `<latent_think>` and `</latent_think>` as boundaries. At latent step s,

$$h_s = f_\theta(e_s, \mathcal{C}_{<s}), \qquad e_{s+1}^{\text{lat}} = g_\phi(h_s), \tag{5}$$

where f_θ is the transformer, $\mathcal{C}_{<s}$ contains the prompt and all preceding latent positions. Each transition extends the same causal cache that will later condition explicit decoding.

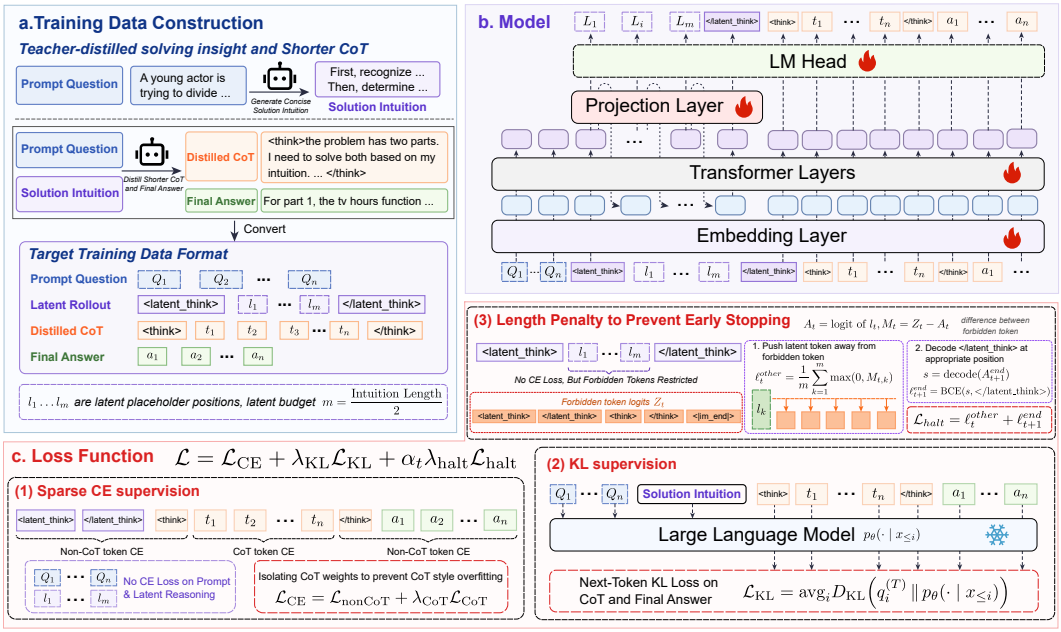

Figure 2: **Supervised LaTER pipeline.** We construct latent-reasoning training sequences with latent and explicit reasoning segments, train the model with supervised and teacher-matching objectives, and obtain a model that first performs latent reasoning and then switches to explicit CoT.

3.2 Intuition-guided data and latent budgets

We construct LATENT-SWITCH-69K from Dolci-Think-SFT-32B reasoning traces (Olmo et al., 2026) in two stages (Figure 2). First, a data-generation teacher extracts a few sentences describing the high-level solution intuition. Then, the teacher receives the problem and intuition and produces a shorter continuation, with instructions to avoid restarting the solution or repeating its plan. Each of the 69,745 retained records contains the problem, intuition, derivation, and answer.

Figure 3 shows the resulting length distributions. The distilled/original CoT length ratio has mean 0.612 and median 0.569. An intuition of L tokens assign approximately $L/2$ latent positions, and a maximum training budget is 128. Assigned budgets have a mean of 41.49 and a median of 40. The intuition also supplies context to the reference teacher used for KL matching below. It therefore determines a latent budget and informs continuation supervision without requiring token-by-token alignment with latent states. Appendix B details corpus construction, statistics, and model inputs. The response format is

$$\underbrace{\texttt{<latent_think>}\ \ell_{1:m}\ \texttt{</latent_think>}}_{\text{latent segment}}\ \ \underbrace{\texttt{<think>}\ t_{1:n}\ \texttt{</think>}\ a}_{\text{explicit continuation}}. \tag{6}$$

3.3 Learning from the explicit continuation

The continuation supplies learning signals to the latent prefix through the causal computation. Prompt and latent-interior positions are masked from ordinary token-level cross-entropy (CE). Let $\mathcal{S}_{\text{CoT}}$ contain interior reasoning positions and $\mathcal{S}_{\text{nonCoT}}$ the remaining supervised positions, including boundary markers and answers. For target-aligned loss $c_i = -\log p_{\theta,\phi}(y_i \mid x_{<i})$, we use

$$\mathcal{L}_{\text{CE}} = \frac{\sum_{i\in\mathcal{S}_{\text{nonCoT}}} c_i}{|\mathcal{S}_{\text{nonCoT}}|} + \lambda_{\text{CoT}}\frac{\sum_{i\in\mathcal{S}_{\text{CoT}}} c_i}{|\mathcal{S}_{\text{CoT}}|}, \quad \lambda_{\text{CoT}} = 0.5. \tag{7}$$

This weighting aims to prevent excessively long chains of thought and repetitive reflection cues such as "Wait..." from dominating training and drawing the model into unproductive loops.

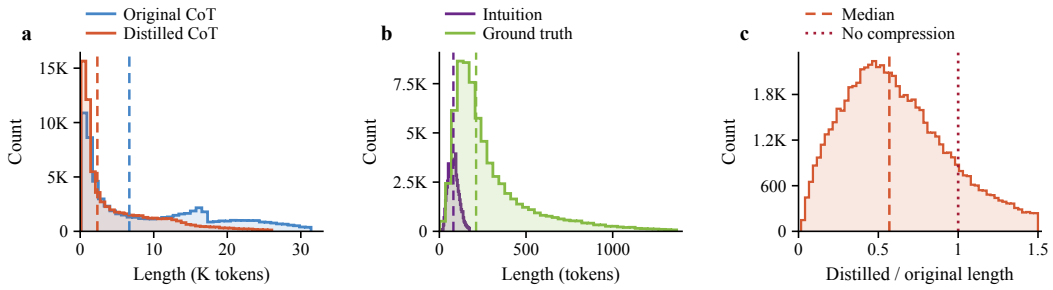

Figure 3: **Training-corpus lengths.** **a**, Original and distilled CoT. **b**, Intuition and ground-truth answer. **c**, Per-example distilled/original length ratios. Dashed lines mark medians.

A frozen copy of the initialization model receives the problem and intuition and is teacher-forced on the shortened continuation. This reference teacher provides cached top-128 distributions for matching at aligned explicit positions $\mathcal{S}_{\mathrm{KL}}$:

$$\mathcal{L}_{\mathrm{KL}} = \frac{1}{|\mathcal{S}_{\mathrm{KL}}|} \sum_{i \in \mathcal{S}_{\mathrm{KL}}} D_{\mathrm{KL}}(q_i \,\|\, p_{\theta,\phi}(\cdot \mid x_{<i})) \,. \tag{8}$$

The temperature is 1.0 and $\lambda_{\mathrm{KL}} = 0.25$. This privileged-context supervision is related to Self-Distilled Reasoner (Zhao et al., 2026), while our KL matching uses cached teacher distributions on fixed reference traces. Both CE and KL supervise the projector by following explicit positions.

3.4 Aligning the Training and Inference Boundaries

The model also needs to learn when to return to text. During training, we execute all m prescribed latent steps even if the model predicts the closing token early. At inference, that prediction immediately ends the latent phase. Because CE is masked inside the latent segment, it does not directly penalize these premature stops. We therefore add a halting objective to discourage early exits and supervise the intended transition.

Let $\mathcal{I}$ denote latent-interior prediction positions and $\mathcal{J}$ the positions supervised for stopping, including the target boundary. The target b_i is one only at that boundary. With vocabulary logits $z_{i,v}$, forbidden structural-token set $\mathcal{V}_{\mathrm{forbid}}$, and maximum allowed-token logit $z^{\mathrm{allow}}_{i,\max}$, we use

$$\mathcal{L}_{\mathrm{halt}} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sum_{v \in \mathcal{V}_{\mathrm{forbid}}} [z_{i,v} - z^{\mathrm{allow}}_{i,\max}]_+ + \frac{1}{|\mathcal{J}|} \sum_{i \in \mathcal{J}} \mathrm{BCE}(\sigma(z_{i,\mathrm{stop}}), b_i) \,, \tag{9}$$

where $z_{i,\mathrm{stop}}$ is the logit of `</latent_think>`. The first term suppresses premature structural-token predictions. The second supervises the exit target through a sigmoid of the raw stop logit. The combined objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}} + \lambda_{\mathrm{KL}} \mathcal{L}_{\mathrm{KL}} + \alpha_t \lambda_{\mathrm{halt}} \mathcal{L}_{\mathrm{halt}}. \tag{10}$$

The base weight is $\lambda_{\mathrm{halt}} = 0.025$; the gate α_t downweights halting when current CE exceeds its moving average (Appendix B.5). Unlike continuation CE and KL, $\mathcal{L}_{\mathrm{halt}}$ directly supervises latent-position logits. At inference, LaTER-SFT advances until it predicts `</latent_think>`, then generates the explicit continuation with the accumulated cache.

4 Experiments

4.1 Experimental Setup

We evaluate seven benchmarks spanning mathematics, science, and knowledge reasoning, and code generation. Supervised experiments use Qwen3-8B and Qwen3-14B (Yang et al., 2025) and training-free experiments additionally cover DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025) and OLMo-3-32B-Think(Olmo et al., 2026). CoT denotes each model's original explicit reasoning baseline. All trained methods use the same Latent-Switch-69K. We report accuracy and Tokens, where LaTER's generation length $C = S + N_{\mathrm{explicit}}$ counts latent steps plus explicit reasoning and answer tokens. Appendix A provides benchmarks, baselines, metrics, and evaluation protocols.

Table 1: **Supervised results on Qwen3-8B and Qwen3-14B.** CoT-SFT uses the same training corpus as LaTER-SFT. Acc. is accuracy (%); Tokens is generation length (C), including latent steps for LaTER. Bold marks the highest accuracy among the methods; blue highlights LaTER-SFT.

	CoT		CoT-SFT		LaTER-SFT	
Benchmark	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens
Qwen3-8B						
AIME 2025	53.3	14,672	60.0	12,790	**63.3**	11,378
GSM8K	88.8	1,153	93.2	1,124	**93.6**	872
MATH-500	89.2	4,144	**90.2**	3,966	89.4	2,909
ARC-Challenge	91.0	892	**94.9**	875	91.4	653
GPQA Diamond	47.0	6,039	49.5	5,954	**52.0**	4,284
HumanEval+	86.6	2,809	**89.0**	2,710	87.8	2,391
MBPP+	75.7	2,635	77.2	2,597	**78.3**	2,228
Mean acc. (%)	75.9	–	79.1	–	**79.4**	–
Mean ΔC (%)	–	0.0	–	−4.0	–	−23.3
Qwen3-14B						
AIME 2025	70.0	15,730	73.3	12,687	**80.0**	10,575
GSM8K	92.7	879	94.1	743	**94.8**	644
MATH-500	93.4	3,472	94.0	3,014	**96.4**	2,541
ARC-Challenge	96.3	631	95.5	578	**96.7**	434
GPQA Diamond	62.6	6,301	61.1	5,885	**63.1**	3,885
HumanEval+	90.8	2,427	90.2	2,071	**92.2**	1,776
MBPP+	82.0	2,144	78.0	1,817	**82.8**	1,717
Mean acc. (%)	84.0	–	83.7	–	**86.6**	–
Mean ΔC (%)	–	0.0	–	−13.3	–	−28.9

4.2 Supervised experiment results

Qwen3-8B. LaTER-SFT achieves higher accuracy with fewer generated tokens than CoT on all seven benchmarks (Table 1), reaching 79.4% mean accuracy with a mean relative token reduction of 23.3%. Compared with CoT-SFT, it improves accuracy on four benchmarks, raises mean accuracy by 0.3 percentage points, and reduces tokens on every benchmark, averaging 19.9% savings. On GPQA Diamond, it gains 2.5 percentage points while reducing tokens by 28.0%.

Against supervised latent-reasoning methods (Table 2), LaTER-SFT achieves the highest accuracy on every benchmark. Its mean accuracy exceeds the strongest of these baselines, SIM-CoT, by 16.8 percentage points. The advantage over the next-best method reaches 36.6 points on AIME 2025 and 20.1 points on HumanEval+. These comparisons support retaining an explicit continuation after the latent prefix to achieve high reasoning accuracy.

Qwen3-14B. LaTER-SFT improves accuracy and reduces tokens relative to both explicit baselines on all seven benchmarks. Against CoT, mean accuracy rises from 84.0% to 86.6%, with a mean relative token reduction of 28.9%. On AIME 2025, accuracy increases from 70.0% to 80.0% while tokens fall from 15,730 to 10,575. Against CoT-SFT, mean accuracy improves by 2.8 percentage points with a mean relative token reduction of 17.8%. The same-data comparison shows that these gains persist when the explicit baseline is also adapted on the constructed corpus.

4.3 Training-free results across methods and backbones

On Qwen3-8B, LaTER-TF improves accuracy over CoT on six tasks and reduces tokens on all seven (Table 3), yielding 78.0% mean accuracy and 17.8% mean token savings. It achieves the highest mean accuracy among the four compared methods. On MATH-500, accuracy improves by 2.8 percentage points while tokens decrease by 19.4%. These gains require no additional training.

Across backbones (Table 4), LaTER-TF improves Qwen3-14B mean accuracy by 0.5 percentage points with 23.5% mean token savings, reducing tokens on all seven benchmarks. Benefits also extend beyond Qwen: on GSM8K, DeepSeek and OLMo gain 1.9 and 2.2 accuracy points while

Table 2: Supervised latent-reasoning methods on Qwen3-8B. All methods use LATENT-SWITCH-69K. ΔC is the relative change from the CoT baseline in Table 1.

	Coconut		CODI		SIM-CoT		LaTER-SFT	
Benchmark	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens
AIME 2025	26.7	765	23.3	217	23.3	3,359	**63.3**	11,378
GSM8K	77.1	375	78.4	214	84.8	614	**93.6**	872
MATH-500	68.0	625	51.0	219	78.4	1,580	**89.4**	2,909
ARC-Challenge	82.7	260	80.7	52	75.8	557	**91.4**	653
GPQA Diamond	36.9	400	32.8	144	45.5	1,463	**52.0**	4,284
HumanEval+	67.1	539	62.2	123	67.7	1,307	**87.8**	2,391
MBPP+	57.1	463	61.6	102	62.7	1,122	**78.3**	2,228
Mean acc. (%)	59.4	–	55.7	–	62.6	–	**79.4**	–
Mean ΔC (%)	–	−82.1	–	−94.0	–	−58.6	–	−23.3

Table 3: Training-free methods on Qwen3-8B. LatentSeek and LTPO optimize latent states at test time. LaTER-TF uses entropy and terminating-token probes to determine when to switch to explicit reasoning. LaTER-TF achieves the highest mean accuracy and shorter generations than CoT.

	CoT		LatentSeek		LTPO		LaTER-TF	
Benchmark	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens
AIME 2025	53.3	14,672	23.3	1,113	43.3	15,761	**56.7**	12,777
GSM8K	88.8	1,153	72.8	799	**92.9**	833	91.1	945
MATH-500	89.2	4,144	70.2	1,041	84.4	3,115	**92.0**	3,342
ARC-Challenge	91.0	892	77.9	706	89.1	760	**92.2**	695
GPQA Diamond	47.0	6,039	**50.0**	5,742	41.4	5,038	49.0	4,912
HumanEval+	**86.6**	2,809	20.1	1,044	81.1	3,565	85.4	2,416
MBPP+	75.7	2,635	40.5	972	77.2	3,150	**79.6**	2,126
Mean acc. (%)	75.9	–	50.7	–	72.8	–	**78.0**	–
Mean ΔC (%)	–	0.0	–	−50.0	–	−4.3	–	−17.8

Table 4: Training-free results across model families and sizes. Each block compares CoT and LaTER-TF on the same backbone, testing the applicability of probe-based switching beyond Qwen.

	AIME 2025		GSM8K		MATH-500		ARC-Challenge		GPQA Diamond		Human Eval+		MBPP+	
Method	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens
DeepSeek-R1-Distill-Llama-8B														
CoT	**33.3**	18,546	76.7	498	**82.6**	2,932	**87.7**	844	39.9	6,186	**76.2**	3,890	67.1	3,154
LaTER-TF	30.0	11,622	**78.6**	378	79.8	2,927	85.6	983	**41.9**	5,896	73.1	3,051	**69.3**	2,351
Qwen3-14B														
CoT	70.0	15,730	92.7	879	93.4	3,472	96.3	631	**62.6**	6,301	**90.8**	2,427	**82.0**	2,144
LaTER-TF	**73.3**	10,661	**93.2**	649	**97.2**	2,887	**96.5**	533	60.6	4,445	**90.8**	1,790	79.6	1,760
OLMo-3-32B-Think														
CoT	**60.0**	17,042	94.3	971	**97.6**	4,197	95.0	791	**61.6**	10,460	**89.0**	3,260	**82.5**	2,760
LaTER-TF	**60.0**	13,251	**96.5**	882	93.6	3,427	**95.3**	882	60.1	7,145	**89.0**	3,201	80.9	2,710

reducing tokens by 24.1% and 9.2%, respectively. Section 5 next examines how transition placement and supervision contribute to the accuracy–cost gains.

5 WHY THE TRANSITION MATTERS

We analyze transition placement and the training objectives that shape the prefix and its exit, then test whether the resulting generation savings translate into faster decoding.

5.1 SENSITIVITY TO TRANSITION PLACEMENT

On Qwen3-14B AIME 2025, adaptive LaTER-TF outperforms every tested fixed horizon (Figure 4). At $\tau_{\mathcal{H}} = 7$, it reaches 73.3% accuracy, 10.0 percentage points above the best fixed-horizon result

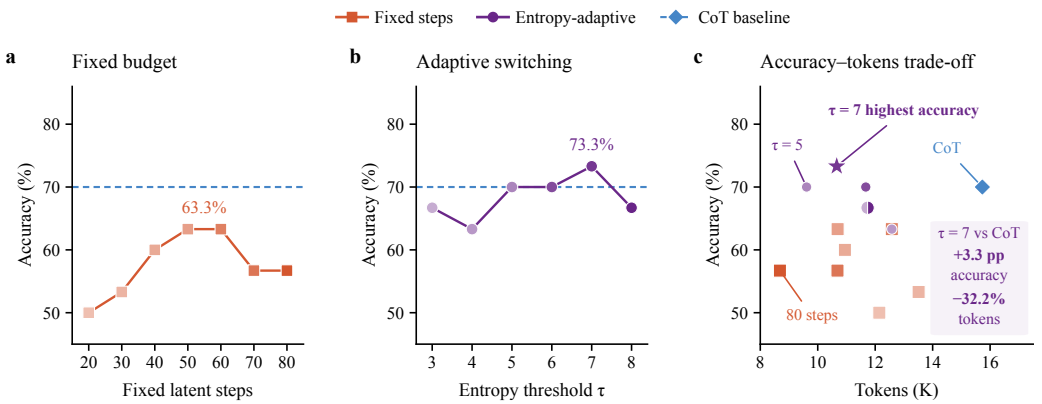

Figure 4: **LaTER-TF switching sensitivity on Qwen3-14B, AIME 2025. a**, Fixed latent budgets. **b**, Entropy thresholds for the adaptive rule, which also retains the terminating-token criterion. **c**, The resulting accuracy–token tradeoffs. Tokens include latent steps and explicit tokens.

Table 5: **Loss ablations on Qwen3-14B.** No halt removes the combined halting objective; no KL removes reference-distribution matching.

	CoT		LaTER-SFT		No halt		No KL	
Benchmark	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens	Acc.	Tokens
AIME 2025	70.0	15,730	**80.0**	10,575	73.3	14,252	76.7	11,659
GSM8K	92.7	879	**94.8**	644	93.9	732	94.3	673
MATH-500	93.4	3,472	**96.4**	2,541	94.4	3,014	95.0	2,721
ARC-Challenge	96.3	631	**96.7**	434	95.3	653	95.7	413
GPQA Diamond	62.6	6,301	**63.1**	3,885	61.6	5,471	57.6	4,337
HumanEval+	90.8	2,427	**92.2**	1,776	90.8	2,294	91.5	1,722
MBPP+	82.0	2,144	82.8	1,717	79.4	2,573	**83.3**	1,864
Mean acc. (%)	84.0	–	**86.6**	–	84.1	–	84.9	–
Mean savings (%)	–	0.0	–	28.9	–	4.9	–	25.5

of 63.3% at 50–60 latent steps. Its 10,661 tokens also represent a 32.2% reduction from CoT. At $\tau_{\mathcal{H}} = 5$, the adaptive rule matches CoT's 70.0% accuracy with approximately 9.6K tokens. These results favor adapting the exit to the trajectory.

Probe entropy varies across examples, so the same threshold can correspond to different latent lengths. Although low entropy indicates concentrated token predictions, it can persist throughout a latent prefix that precedes an incorrect continuation. Appendix C brings together these entropy diagnostics, a failure case, and PCA visualizations of trajectory behavior. We next test whether supervising the transition helps preserve the accuracy–cost gains.

5.2 Effects of Halting and Teacher Matching

Halting supervision improves accuracy and reduces tokens on every benchmark relative to the no-halt variant (Table 5). The full model reaches 86.6% mean accuracy versus 84.1%, and increases mean token savings over CoT from 4.9% to 28.9%. On AIME 2025, it gains 6.7 percentage points while using 3,677 fewer tokens. No-halt traces predict `</latent_think>` after 3–5 latent steps: training completes the prescribed rollout, whereas inference exits at this prediction. Explicit boundary supervision addresses this mismatch and improves the complete rollout, including the continuation generated after the exit.

KL matching further improves accuracy on six benchmarks, raising mean accuracy from 84.9% to 86.6% and mean token savings over CoT from 25.5% to 28.9%. Its largest accuracy gain is on GPQA Diamond, from 57.6% to 63.1%, followed by a 3.3-point gain on AIME 2025. These results support the complementary roles of the two objectives: halting directly trains the exit behavior, while teacher matching guides the explicit continuation.

5.3 DECODING TIME

On Qwen3-8B, LaTER-SFT decodes faster than both explicit baselines on all seven benchmarks using 2*NVIDIA H100 GPUs and batch size 15 (Table 6). Mean per-benchmark time reductions are 19.1% versus CoT and 14.6% versus CoT-SFT.

Table 6: Decoding time (s) on Qwen3-8B. LaTER-SFT timings include latent computation. Bold marks the lowest time.

Benchmark	CoT	CoT-SFT	LaTER-SFT
AIME 2025	255.43	228.73	**219.49**
GSM8K	12.61	12.29	**9.37**
MATH-500	64.86	62.38	**55.36**
ARC-Challenge	12.59	10.74	**7.78**
GPQA Diamond	101.73	98.72	**86.34**
HumanEval+	51.12	49.65	**45.44**
MBPP+	50.25	49.81	**42.64**

ARC-Challenge shows the largest reductions: 38.2% versus CoT and 27.6% versus CoT-SFT. These measurements confirm a net decoding-time benefit after accounting for latent computation. Appendix A.4 gives the timing protocol, and training costs appear in Appendix B.6.

6 RELATED WORKS

Training-free latent reasoning. Training-free methods vary in how they construct continuous inputs and interleave them with text. Soft Thinking feeds back probability-weighted mixtures of token embeddings (Zhang et al., 2025), while SeLaR activates soft embeddings at uncertain steps through entropy gating, applying latent reasoning only at high-uncertainty steps and preserving discrete decoding at deterministic steps(Fu & Luo, 2026). LatentMAS projects hidden states into the input embedding space and shares latent working memory across agents (Zou et al., 2026). LaTER-TF follows this projection for its recurrent latent prefix. SwiReasoning alternates latent and explicit blocks using entropy trends (Shi et al., 2026). LaTER makes a single transition to an explicit continuation, retaining the prefix's KV cache and treating the transition boundary as a design variable.

Training-based latent reasoning. Coconut feeds final-layer hidden states back as input embeddings, supporting autoregressive latent reasoning and implicit breadth-first exploration (Hao et al., 2025). SoftCoT learns a projector mapping assistant-generated soft thoughts into a frozen target model's representation space (Xu et al., 2025). Other methods refine these representations through structural constraints or additional supervision. Latent-SFT constrains latent states to the vocabulary column space (Deng et al., 2025), while CoLaR combines compressed-embedding prediction with reinforcement learning (Tan et al., 2025). CODI transfers explicit CoT into latent reasoning through hidden-state self-distillation (Shen et al., 2025), whereas SIM-CoT uses an auxiliary decoder to supervise individual latent steps (Wei et al., 2026). These approaches focus primarily on compressing explicit reasoning into continuous representations. LaTER-SFT trains a latent prefix to prepare a shortened explicit continuation and jointly supervises when to return to text. This exit supervision addresses the mismatch between prescribed training rollouts and early inference exits.

7 CONCLUSION

We introduce LaTER, a latent-then-explicit reasoning paradigm for reducing test-time token cost without discarding explicit verification. The method separates reasoning into two phases: a continuous latent rollout for early exploration, followed by discrete CoT generation for symbolic checking and final-answer construction. In the training-free setting, we have found that pretrained reasoning models already exhibit structured latent trajectories, including terminating-token probes and informative entropy dynamics. A simple adaptive switch based on these signals reduces token usage and can improve accuracy on several benchmarks. We then construct LATENT-SWITCH-69K and train a LaTER model with a learned latent projector and halting supervision. The trained model improves the accuracy–efficiency tradeoff across mathematics, coding, and knowledge-intensive benchmarks, while using one third fewer tokens than standard CoT.

LaTER also leaves open important questions. The training-free switch is still a hand-crafted approximation, and the trained model's behavior depends on the latent-budget distribution and the quality of distilled intuitions. Future work should learn richer instance-adaptive halting policies, study longer latent exploration for open-ended tasks, and extend latent-then-explicit reasoning to multimodal settings where full verbalization can be especially costly.

AI USE STATEMENT

We used generative AI tools to assist with manuscript preparation, primarily for grammar correction and language polishing. As part of the experimental methodology, we also used an LLM to construct the training corpus from Dolci-Think-SFT-32B, as described in Section 3.2 and Appendix B.1. In addition, AI-assisted coding tools were used to help implement portions of the experimental code. We reviewed and verified all AI-assisted text, data-processing outputs, and code, and take full responsibility for the final manuscript, experimental results, and associated artifacts.

ETHICS STATEMENT

This work develops and evaluates reasoning methods for language models using existing pretrained models and publicly available research datasets. It does not involve human subjects, direct user studies, or the collection of new personal data. We follow the licenses and usage conditions of the models, datasets, and software used in our experiments, as discussed in Appendix E. Continuous latent reasoning states are less directly inspectable than textual reasoning traces and may therefore make model behavior more difficult to audit. The models and methods presented here are intended for research and should not be used in high-stakes applications without independent validation and appropriate safeguards.

REPRODUCIBILITY STATEMENT

Sections 2 and 3 define the inference interface and training objectives, and Section 4 reports the main comparisons. Appendix A collects benchmarks, evaluation metrics, decoding settings, baseline configurations, and ablation and timing protocols. Appendix B describes corpus construction and training implementation; Appendix C provides switching and trajectory diagnostics. Prompt templates and external assets are documented in Appendices D and E, respectively. Detailed experimental code, runnable configurations, and additional hyperparameters are provided in the anonymous code repository to support reproduction of the reported results.

REFERENCES

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, 2025. URL https://arxiv.org/abs/2505.23281.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. URL https://arxiv.org/abs/2307.08691.

Jingcheng Deng, Liang Pang, Zihao Wei, Shicheng Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. Latent reasoning in llms as a vocabulary-space superposition, 2025. URL https://arxiv.org/abs/2510.15522v1.

Renyu Fu and Guibo Luo. SeLaR: Selective latent reasoning in large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.320. URL https://aclanthology.org/2026.acl-long.320/.

Daya Guo, Dejian Yang, Haowei Zhang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, September 2025. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

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.

Hengli Li, Chenxi Li, Tong Wu, Xuekai Zhu, Yuxuan Wang, Zhaoxin Yu, Eric Hanchen Jiang, Song-Chun Zhu, Zixia Jia, Ying Nian Wu, and Zilong Zheng. Seek in the dark: Reasoning via test-time instance-level policy gradient in latent space, 2026. URL https://arxiv.org/abs/2505.13308.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and LINGMING ZHANG. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2023. doi: 10.52202/075280-0943. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/43e9d647ccd3e4b7b5baab53f0368686-Paper-Conference.pdf.

Jiawei Liu, Songrun Xie, Junhao Wang, Yuxiang Wei, Yifeng Ding, and Lingming Zhang. Evaluating language models for efficient code generation. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=IBCBMeAhmC.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Team Olmo, Allyson Ettinger, Amanda Bertsch, et al. Olmo 3, 2026. URL https://arxiv.org/abs/2512.13961.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

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*, Suzhou, China, November 2025. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

Dachuan Shi, Abedelkadir Asi, Keying Li, Xiangchi Yuan, Leyan Pan, Wenke Lee, and Wen Xiao. Swireasoning: Switch-thinking in latent and explicit for pareto-superior reasoning LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=t33kMzEAg8.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of llm reasoning chains. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2025. doi: 10.52202/085713-0164. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0706261aedab63814a2b73c32564b4c4-Paper-Conference.pdf.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

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 *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=6YRJ4jmVQl.

Junde Wu, Jiayuan Zhu, Fengling Liu, Minhao Hu, and Jiazhen Pan. J-cot: Chain-of-thought in j-space, 2026. URL https://arxiv.org/abs/2607.21981.

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)*, 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, et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Wengao Ye, Yan Liang, and Lianlei Shan. Thinking on the fly: Test-time reasoning enhancement via latent thought policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=r1WEQzkCQv.

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. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2025. doi: 10.52202/085713-5629. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f7396d1c54d51416958d63e285377103-Paper-Conference.pdf.

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.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=syG9I9ofd8.

A REPRODUCIBILITY

This appendix collects the settings for the comparisons in Section 4 and the ablation and timing studies in Section 5.

A.1 BENCHMARKS AND MODELS

We evaluate mathematics (AIME 2025, MATH-500, GSM8K), knowledge-intensive and science reasoning (ARC-Challenge, GPQA Diamond), and code generation (HumanEval+, MBPP+) (Balunović et al., 2025; Lightman et al., 2023; Cobbe et al., 2021; Clark et al., 2018; Rein et al., 2024; Liu et al., 2023; 2024). The training-free study covers Qwen3-8B, Qwen3-14B, DeepSeek-R1-Distill-Llama-8B, and OLMo-3-32B-Think (Yang et al., 2025; Guo et al., 2025; Olmo et al., 2026) and supervised experiments use both Qwen3 sizes.

We compare **CoT** (the original model with explicit reasoning), **CoT-SFT** (a same-data fine-tuning control without latent computation or halting loss), **LaTER-TF** (with probe-based switching), and **LaTER-SFT** (with learned projection and stopping). Additional baselines include LatentSeek and LTPO (Li et al., 2026; Ye et al., 2026), and supervised Coconut, CODI, and SIM-CoT (Hao et al., 2025; Shen et al., 2025; Wei et al., 2026). All training methods use the same training corpus LATENT-SWITCH-69K.

A.2 METRICS AND DECODING PROTOCOL

Generation accounting. We count latent transitions and emitted tokens as generation positions:

$$C_{\mathrm{LaTER}} = S + N_{\mathrm{explicit}}, \qquad C_{\mathrm{CoT}} = N_{\mathrm{CoT}}. \tag{11}$$

The explicit count includes reasoning and answer tokens. A latent prefix reduces this count only when the explicit tokens it saves exceed the latent positions it adds. Tables report the mean count per benchmark as Tokens. Section 5.3 reports measured decoding time separately.

Decoding settings. Benchmark evaluations use random seed 42. Methods whose original protocols require greedy decoding retain it; sampling-based evaluations use temperature 0.6, top-$p = 0.95$, and top-$k = 20$. Appendix D.1 gives the evaluation prompts.

Following the benchmark-specific generation limits used by LatentMAS (Zou et al., 2026), we cap generation at 2,048 positions for GSM8K and ARC-Challenge, 4,096 for HumanEval+ and MBPP+, 8,192 for MATH-500 and GPQA Diamond, and 20,000 for AIME 2025. For LaTER, these limits count both latent steps and explicit tokens.

Training-free probe entropy is computed from raw logits over the full vocabulary using natural logarithms, as in Equation 3. Sampling filters are not applied to the probe distribution.

A.3 COMPARISON-METHOD CONFIGURATIONS

Training-free methods. LatentSeek uses at most 10 test-time optimization steps, Adam learning rate 0.05, and reward threshold 0.2. LTPO uses 20 steps, Adam learning rate of 0.05, Gaussian-noise scale $\sigma = 0.1$ with decay factor 0.99, and selection of the state with the highest observed reward.

Supervised methods. We follow the baseline-training protocols reported in J-CoT (Wu et al., 2026) and SIM-CoT (Wei et al., 2026), using LATENT-SWITCH-69K as the common adaptation corpus for all trained methods. CoT-SFT uses the same data and optimization strategy as LaTER-SFT but removes the latent segment, projector, and halting loss. Coconut starts from CoT initialization and uses curriculum training with learning rate 2×10^{-5}, three epochs per stage, and two latent slots per stage. SIM-CoT starts from the resulting Coconut checkpoint and uses learning rate 2×10^{-5}. CODI is trained for 10 epochs with learning rate 10^{-4}, LoRA rank 128, scaling factor 32, dropout 0.1, and projection dimension 2,048.

A.4 Ablation and timing protocols

Loss ablations. Table 5 reports the complete Qwen3-14B ablation. The no-halt variant removes both components of Equation 9; the no-KL variant removes reference-distribution matching while retaining CE and halting. Appendix B.5 describes the forced-rollout behavior that can hide early stop predictions during training.

Decoding-time measurements. Table 6 reports Qwen3-8B measurements on 2*NVIDIA H100 GPUs with batch size 15. The explicit baselines use `model.generate()`. Relative to CoT, reductions range from 11.1% to 38.2%, averaging 19.1%; relative to CoT-SFT, they range from 4.0% to 27.6%, averaging 14.6%.

B Training Data and Implementation

We first describe the construction of Latent-Switch-69K and its conversion into model inputs, then detail continuation supervision, halting, and optimization.

B.1 From source traces to supervised records

We sample source reasoning traces from Dolci-Think-SFT-32B (Olmo et al., 2026) and construct each record in two stages using DeepSeek-V3.2 (reasoning). The intuition-extraction prompt receives the problem, source outputs, optional ground truth, and metadata. It requests a JSON object containing a task summary, a high-level solution intuition, and any incorrect ideas explicitly evidenced in the source outputs. Proposed intuitions contain 2–10 sentences and exclude detailed derivations and final answers. The prompt also records source-answer correctness and the presence of reflection.

The continuation prompt then receives the problem and selected intuition. It asks the teacher to continue from that intuition, keep reasoning compact, and avoid repeated planning. It explicitly allows the supplied intuition to be correct or incorrect. The retained record contains four parts: the problem, intuition, shortened derivation, and final answer. Appendix D.2 reproduces the extraction and continuation prompts.

B.2 Source composition and corpus statistics

The retained corpus contains 69,745 examples. Figure 5 reports all 13 source components and their counts, which sum to the corpus total. OpenThoughts3 Math and Python Algorithms are the two largest components. These source labels identify provenance; they do not supply an exhaustive example-level domain taxonomy.

Table 7: Corpus statistics. Difficulty shares are computed from the retained counts. Length ratios and assigned latent budgets are summarized per example.

Difficulty	Count	Share (%)
Easy	6,667	9.56
Medium	45,650	65.45
Hard	17,428	24.99
Total	69,745	100.00
Statistic	**Mean**	**Median**
Distilled / original CoT length	0.612	0.569
Assigned latent positions	41.49	40.00

Table 7 describes the difficulty mixture and length summaries. Figure 3 compares the original and distilled CoT distributions, intuition and answer lengths, and per-example compression ratios.

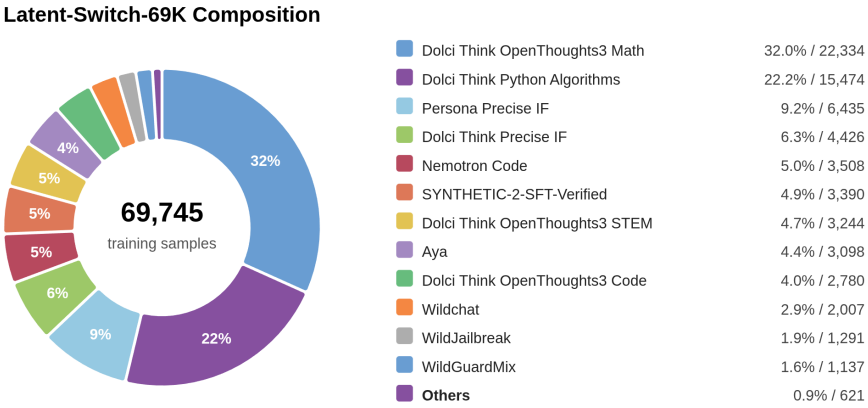

Figure 5: **Source composition of LATENT-SWITCH-69K.** Legend entries give the percentage and count of retained examples from each source component, totaling 69,745 examples.

B.3 LATENT BUDGETS, MODEL INPUTS, AND RECURRENCE

For retained intuition length L, preprocessing assigns an integer budget of approximately $L/2$, subject to a maximum of 128 latent positions and tokenization constraints. Its mean of 41.49 and median of 40 describe the training targets.

The student response follows Equation 6, with repeated placeholders between `<latent_think>` and `</latent_think>`. The explicit segment contains the shortened derivation inside `<think>` and `</think>`, followed by the answer and `<|im_end|>`. Batch construction aligns the assistant prefix and the transition into `<think>` across examples.

For Qwen3-8B and Qwen3-14B, the latent projector is a two-layer MLP with GELU activation. Its input and output dimensions equal the backbone hidden size: 4,096 and 5,120, respectively. The model processes the prompt and executes a cache-based latent rollout using Equation 5. The projected embeddings replace the latent placeholders in the full input sequence. A subsequent causal teacher-forcing pass uses these embeddings inside the latent segment and ordinary token embeddings elsewhere.

The intuition therefore serves two roles: allocating the prescribed latent budget and providing privileged context to the reference teacher. Neither role requires the student to reconstruct intuition text at individual latent positions.

B.4 CONTINUATION SUPERVISION AND REFERENCE MATCHING

Prompt and latent-interior CE labels are masked with -100. The CoT mask contains only tokens strictly inside `<think>` and `</think>`. The complementary supervised mask contains the latent opening and closing tags, explicit-thinking boundaries, final answer, and response terminator. Equation 7 normalizes the two supervised groups separately. CE and KL on the explicit continuation backpropagate through the recurrent embeddings, while halting directly supervises vocabulary logits at latent positions.

The reference teacher is a frozen copy of the initialization model. It receives the problem and distilled intuition and is teacher-forced on the shortened reasoning and answer. Its conversation omits the student's latent placeholders. Unlike the data-generation teacher that constructs records beforehand, this teacher supplies distributions for continuation supervision.

We cache the teacher's top-128 distributions and apply Equation 8 at selected, aligned explicit reasoning and answer positions, with temperature 1.0 and weight 0.25. No vocabulary KL is applied to latent-interior placeholders. The aligned-position mask determines the supervised continuation positions; matching the continuation does not require textual targets for the latent states.

B.5 HALTING SUPERVISION AND FORCED ROLLOUTS

Equation 9 combines an interior structural-token penalty with a binary exit target. The margin term compares forbidden structural-token logits with the largest allowed-token logit. The exit target is zero before the prescribed boundary and one at that boundary. Its BCE uses the sigmoid of the raw closing-token logit, rather than the token's vocabulary-softmax probability.

The effective coefficient in Equation 10 combines the halt weight of 0.025 with the CE-dependent gate

$$\alpha_t = \mathrm{clip}\left(\frac{\mathrm{EMA}(\mathcal{L}_{\mathrm{CE}})_t}{\mathcal{L}_{\mathrm{CE},t} + \epsilon}, 0, 1\right). \tag{12}$$

Here $\mathrm{EMA}(\mathcal{L}_{\mathrm{CE}})_t$ is the moving average of CE and ϵ stabilizes the denominator. When current CE exceeds its moving average, the gate reduces the halting contribution; otherwise it can saturate at one. This regulates the relative loss scale.

Training always executes the prescribed budget m, whereas inference exits at the first predicted closing token. If that prediction occurs at $k < m$, the paths differ:

$$\text{Training:} \quad h_1 \to \cdots \to h_k \to \cdots \to h_m \to \text{explicit CoT} \to a,$$

$$\text{Inference:} \quad h_1 \to \cdots \to h_k \xrightarrow{\text{predicted stop}} \text{explicit CoT} \to a.$$

The no-halt traces predict the closing token after only 3–5 latent steps. Masked latent-interior CE leaves those predictions without a direct token-level penalty even though training still completes the longer rollout. The combined halting loss supervises this boundary mismatch.

B.6 OPTIMIZATION AND COMPUTE RESOURCES

We train for one epoch with AdamW (Loshchilov & Hutter, 2019), using $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.01, and gradient clipping at 1.0. The learning rate starts at 10^{-7} and follows a cosine schedule to 10^{-8}.

Each device processes one example per micro-batch, with four gradient-accumulation steps. The maximum sequence length is 24,096, including at most 128 training latent positions. Training uses bf16 precision, gradient checkpointing, FlashAttention-2, and DeepSpeed ZeRO-3 (Dao, 2023; Rajbhandari et al., 2020). The CoT CE weight is 0.5; reference matching and gated halting use the coefficients specified above.

Qwen3-8B and its trained comparison methods use 4*NVIDIA H100 GPUs. Qwen3-14B uses one node with 8*NVIDIA A800 80GB GPUs; LaTER training takes approximately five days, compared with two days for CoT-SFT on the same adaptation data. The evaluation setup uses 2*NVIDIA A800 80GB GPUs, while the decoding-time study uses 2*NVIDIA H100 GPUs.

C TRANSITION AND TRAJECTORY DIAGNOSTICS

This appendix examines how switching signals vary, illustrates a low-entropy failure, and provides complementary visualizations of latent and explicit trajectories.

C.1 SWITCHING SENSITIVITY AND ENTROPY

Follow Equation 3, we decode the latent states into tokens for observation. We find that, early latent states often decode to tokens with little readable content, such as empty strings or repeated newlines "`\n`". As the rollout progresses, the argmax probe often becomes a terminating token such as `<|im_end|>`, `</think>`, or `<|endoftext|>`. Their appearance therefore suggests that latent trajectories retain the model's learned stopping patterns even without discrete token feedback. Rather than simply drifting through hidden space, they often approach states associated with the end of a reasoning trace.

This observation is central to LaTER. If the model internally approaches a state that resembles "ready to stop", then switching to explicit CoT can be aligned with the model's own trajectory and the reasoning patterns it acquired during pretraining, rather than imposed at an unrelated time.

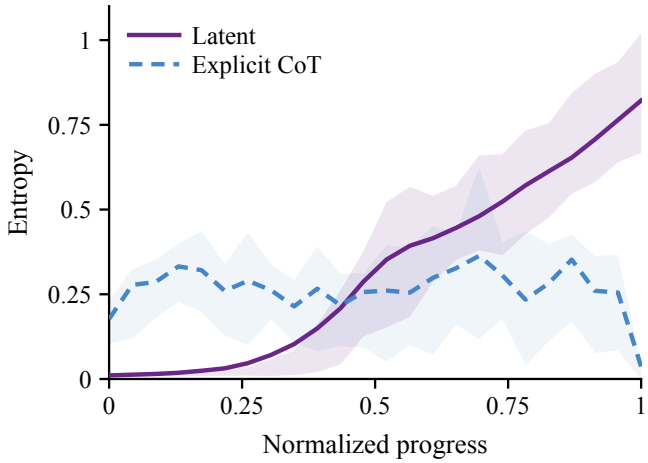

Figure 6: **Normalized-progress entropy on Qwen3-14B AIME 2025.** Lines show means and bands show the 25th–75th percentiles. Latent progress runs from rollout start to termination; explicit progress runs within sentences.

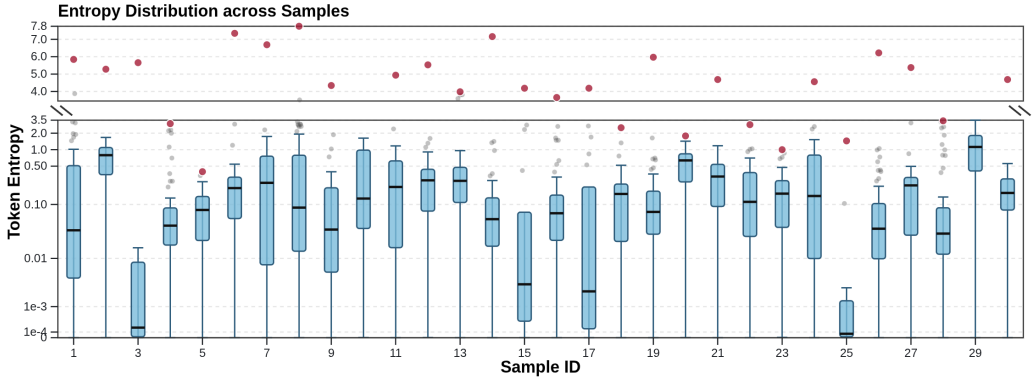

Figure 7: **Within-sample latent entropy distributions on AIME 2025.** Each box groups the recorded token entropies for one sample. Medians, spreads, and upper tails vary across samples. The broken vertical axis retains high-entropy observations alongside the low-entropy distributions.

Entropy over normalized progress. Figure 6 retains a notion of progression by aligning latent rollouts from start to termination. Mean latent entropy rises over the reported normalized trajectory. Explicit CoT is instead aligned within sentences, where the next-token distribution becomes constrained by the preceding words. Entropy alone does not establish how latent reasoning works, but it offers a practical switching signal alongside terminating-token probes. A terminating probe token suggests that the trajectory is approaching a stopping state. High probe entropy indicates greater uncertainty in the decoded distribution.

Fixed and adaptive boundaries. Figure 4 compares two interventions on Qwen3-14B AIME 2025. A fixed horizon assigns the same latent budget to every problem. The adaptive sweep instead varies the entropy threshold while retaining the terminating-token criterion, allowing different examples to exit at different positions. The sweeps establish that boundary placement affects accuracy and total generation length.

Entropy distributions across samples. For each of the 30 AIME 2025 samples, the latent rollout records vocabulary-probe entropy until a model-native terminating token is reached. Figure 7 groups

these token-level observations by sample ID. Each box summarizes positions within one sample's latent trajectory.

The sample distributions span markedly different entropy ranges. A common absolute threshold therefore encounters different parts of each sample's distribution. The case in Section C.2 further shows that low entropy can precede an incorrect continuation.

C.2 A LOW-ENTROPY PREFIX FOLLOWED BY AN INCORRECT DERIVATION

Figure 8 compares explicit CoT and LaTER on AIME 2025 Problem 10.

Failure Case: AIME 2025 Problem 10

Task. The 27 cells of a 3×9 grid are filled in using the numbers 1 through 9 so that each row contains 9 different numbers, and each of the three 3×3 blocks heavily outlined in the example below contains 9 different numbers, as in the first three rows of a Sudoku puzzle.
— 4 — 2 — 8 — 9 — 6 — 3 — 1 — 7 — 5 —
— 3 — 7 — 9 — 5 — 2 — 1 — 6 — 8 — 4 —
— 5 — 6 — 1 — 8 — 4 — 7 — 9 — 2 — 3 —

The number of different ways to fill such a grid can be written as $p^a \cdot q^b \cdot r^c \cdot s^d$, where $p, q, r,$ and s are distinct prime numbers and $a, b, c,$ and d are positive integers. Find $p \cdot a + q \cdot b + r \cdot c + s \cdot d$.

Baseline CoT. The standard explicit CoT run explicitly enumerates the combinatorial factors. It first observes that the first row can be arbitrary (9! choices). Given the first row, it derives 56 possible set-level assignments for the second row and 6^3 within-block orderings. It then notes that the third row still has independent within-block orderings, contributing another 6^3 factor:

$$N_{\text{CoT}} = 9! \cdot 56 \cdot 6^6 = 2^{16}3^{10}5^1 7^2.$$

Therefore,

$$2 \cdot 16 + 3 \cdot 10 + 5 \cdot 1 + 7 \cdot 2 = 81.$$

The final prediction is correct:

$$\text{Pred} = 81, \qquad \text{Gold} = 81.$$

The important qualitative point is that the explicit trace keeps the row/block assignment and ordering factors separate until the final factorization.

Latent reasoning. The output is organized as a latent segment followed by an explicit CoT segment. Abridged from the raw trace:

$\underbrace{\cdots}_{\substack{\text{low-entropy latent tokens} \\ H_{\max}=1.625}}$ `<|endoftext|><think>`

Given Block A, the number of valid Block B's is 56×216 ... Block C is uniquely determined ... Therefore, the total number of grids is $9! \times 56 \times 216$ ... Hence, the final answer is 66.

Thus the explicit CoT after the latent segment follows a superficially similar counting strategy, but it treats the third block as uniquely determined once the first two blocks are chosen. This makes the trace use

$$N_{\text{latent}} = 9! \cdot 56 \cdot 6^3 = 2^{13}3^7 5^1 7^2$$

instead of $9! \cdot 56 \cdot 6^6$. This gives

$$2 \cdot 13 + 3 \cdot 7 + 5 \cdot 1 + 7 \cdot 2 = 66.$$

The final prediction is incorrect:

$$\text{Pred} = 66, \qquad \text{Gold} = 81.$$

Figure 8: **Low-entropy latent reasoning can still fail.** Maximum probe entropy is 1.625 in the latent phase and 3.23 in the explicit continuation. The missing 6^3 factor is visible in the continuation.

C.3 Qualitative hidden-state trajectories

We present sample-level comparisons using both Qwen3-8B and Qwen3-14B on AIME 2025 and GSM8K. Figures 9 and 10 compare standard CoT (Baseline), training-free LaTER, supervised LaTER (SFT), and Coconut with Qwen3-8B. Figures 12 and 13 provide the corresponding Baseline and LaTER comparisons with Qwen3-14B. Together, these plots provide a qualitative view of how the latent and explicit phases are arranged in the hidden-state space across model scales.

Figure 9 shows samples 5, 6, 8, 17, and 30. Baseline forms broad, overlapping explicit trajectories, with PC1 accounting for only 12.4%–24.9% of the variance across the displayed samples. Training-free LaTER consistently begins with a compact blue segment separated from most of the red states, followed by an elongated explicit trajectory; its PC1 fraction is substantially higher and more stable at 34.0%–37.4%. LaTER (SFT) also shows a distinct latent segment, while the explicit states often follow an oblique, elongated band, especially in samples 5, 8, 17, and 30. Its PC1 fraction ranges from 31.1% to 54.1%, reaching 54.1% on sample 30.

The Coconut panels contain latent segments but do not show the same consistent separation between a compact latent cluster and the bulk of the explicit states. For example, in samples 6, 8, and 17, the blue paths traverse or overlap the region occupied by red states. Coconut's explicit trajectories remain broad or fan-shaped, with PC1 fractions of 20.8%–30.7%.

Figure 10 displays samples 4, 49, 115, 295, 386, 557, 690, 828, 1158, and 1291. Baseline again shows overlapping explicit paths, with PC1 fractions of 10.8%–18.3%. Training-free LaTER exhibits a recurring compact latent segment followed by a spatially separated, elongated explicit trajectory, with PC1 consistently accounting for 35.3%–38.7% of the variance. LaTER (SFT) has higher PC1 fractions than training-free LaTER on all ten displayed samples, ranging from 38.4% to 54.6%, although its red trajectories vary substantially: samples 4 and 557 retain broad paths, whereas sample 1291 forms a narrow diagonal band.

Across the two datasets, the most consistent distinction is the visible organization of the latent-to-explicit transition in LaTER.

Figures 12 and 13 show that the qualitative phase structure observed with Qwen3-8B also appears with Qwen3-14B. On AIME 2025, training-free LaTER begins with a visibly distinct latent segment before entering the explicit-state region in all five displayed samples. The GSM8K panels show the same recurring transition: both LaTER variants contain a compact latent segment connected to a predominantly explicit trajectory, while Baseline contains only broad explicit paths.

These visualizations motivate that LaTER not only makes the latent phase compact, but also constrains the subsequent explicit reasoning trajectory to a lower-dimensional region of the final-layer representation space. This interpretation is consistent with LaTER's recurrent latent transition,

$$h_t \longrightarrow e_{t+1}^{\mathrm{lat}} \longrightarrow h_{t+1},$$

and with its reuse of the complete latent KV cache during subsequent explicit CoT. A plausible explanation is that the latent phase primarily forms a global solution state or plan state. If this recurrence suppresses uncertain search directions and selects a solution basin before the switch, the explicit phase need not wander over as broad a representation region as Baseline. Instead, it can remain within the selected basin while performing symbolic expansion, verification, and local correction. The repeated back-and-forth motion of the red LaTER trajectories is consistent with iterative refinement along a limited set of directions, suggesting a division of labor in which the latent phase performs global search and the explicit phase performs local refinement and verification.

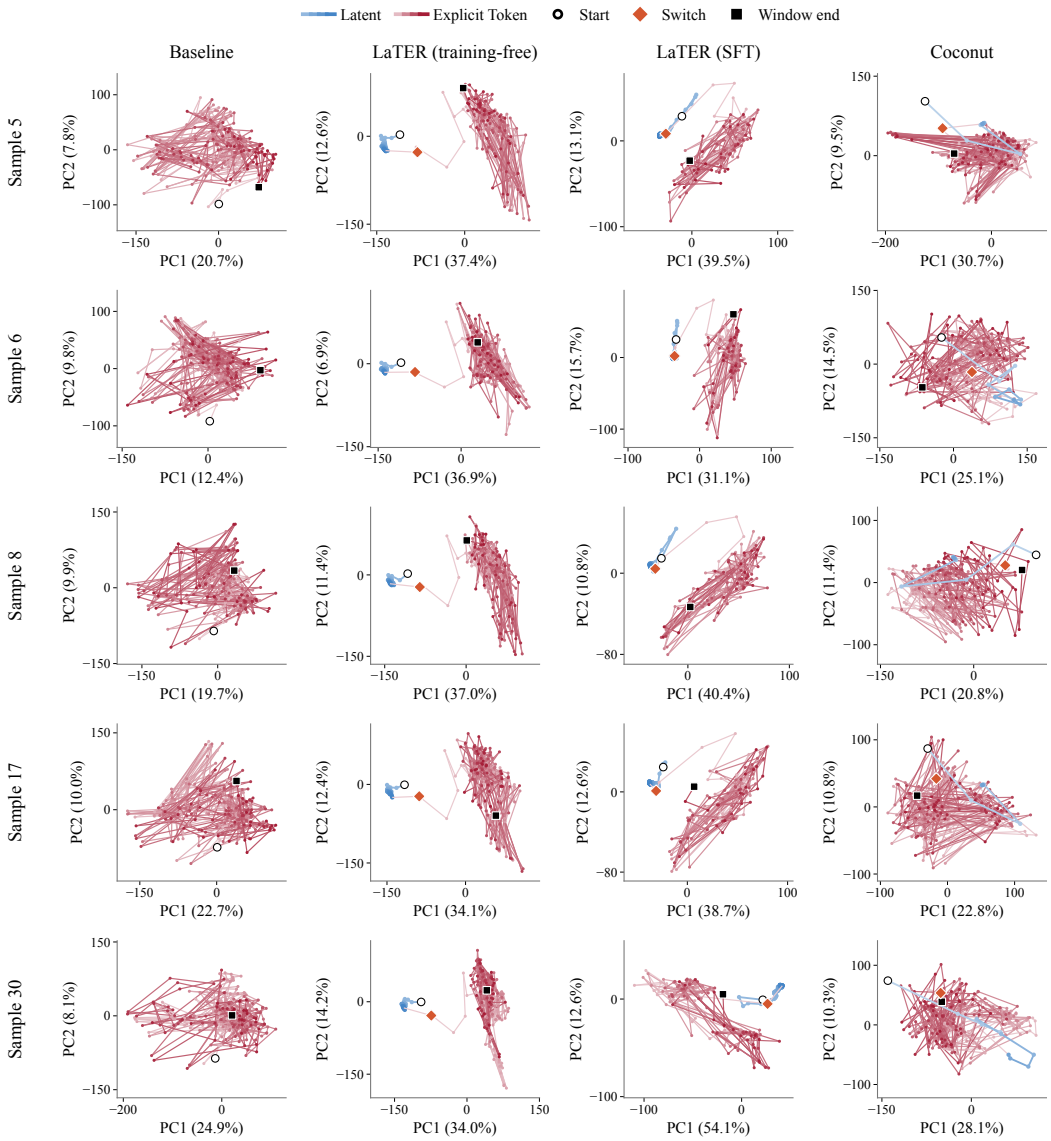

Figure 9: **Qwen3-8B on AIME 2025: samples 5, 6, 8, 17, and 30.** Columns show CoT (Baseline), LaTER-TF, LaTER-SFT, and Coconut. Blue/red denote latent/explicit states; circles, diamonds, and squares mark starts, switches, and display-window ends.

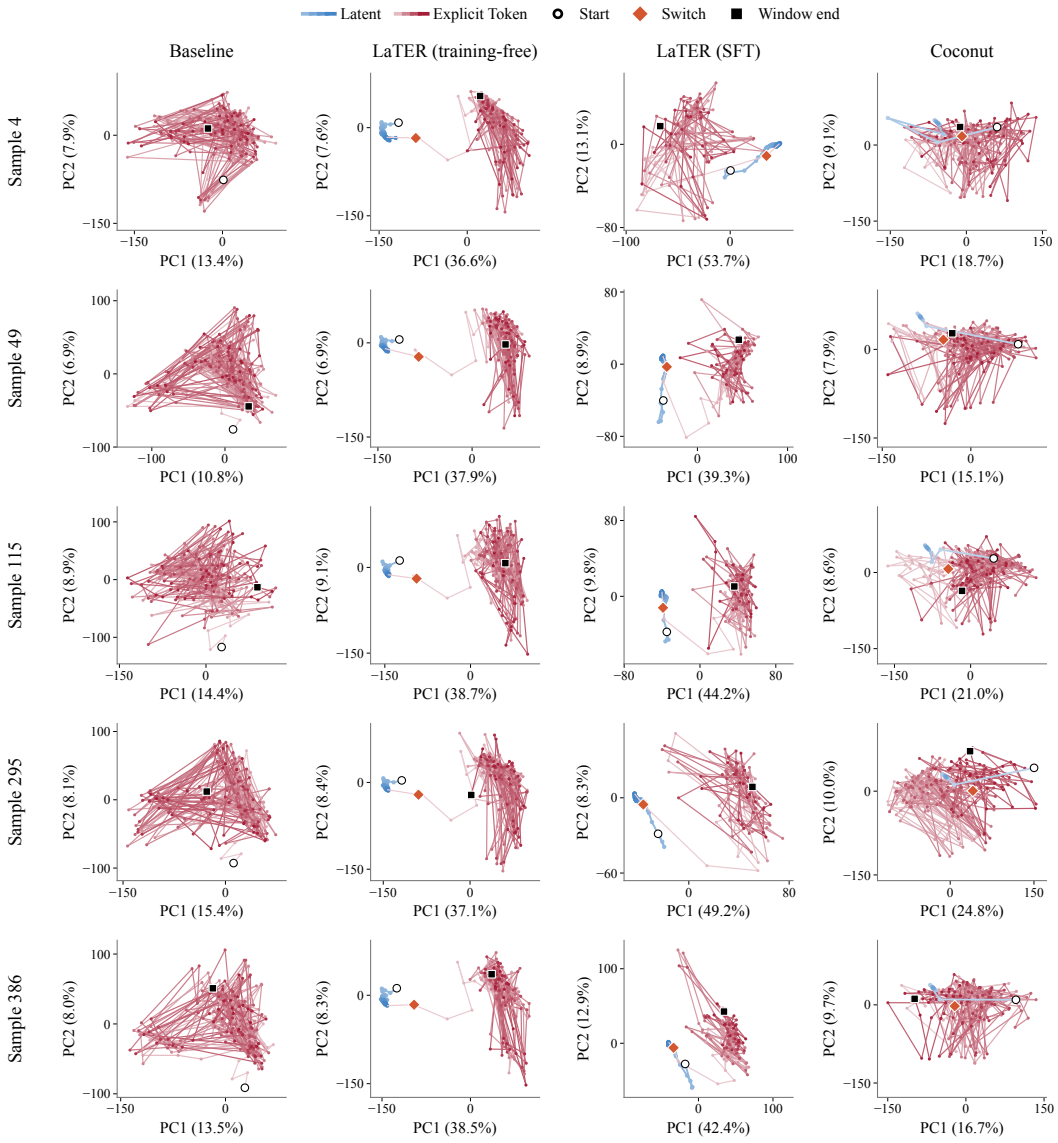

Figure 10: **Qwen3-8B on GSM8K: samples 4, 49, 115, 295, and 386.** Columns show CoT, LaTER-TF, LaTER-SFT, and Coconut, with the same phase colors and markers as Figure 9.

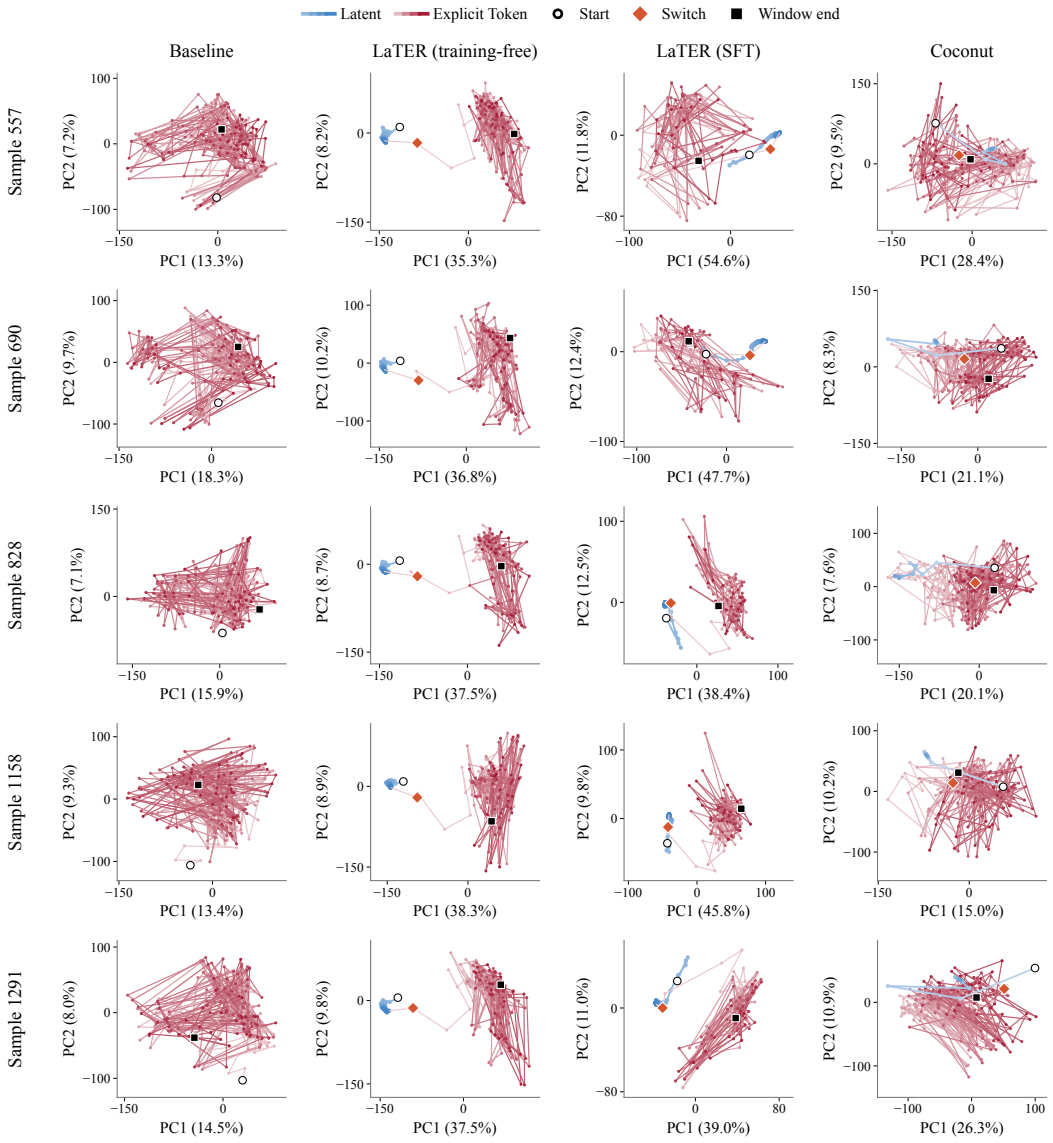

Figure 11: **Qwen3-8B on GSM8K: samples 557, 690, 828, 1158, and 1291.** Continuation of Figure 10, retaining the same method order and display conventions.

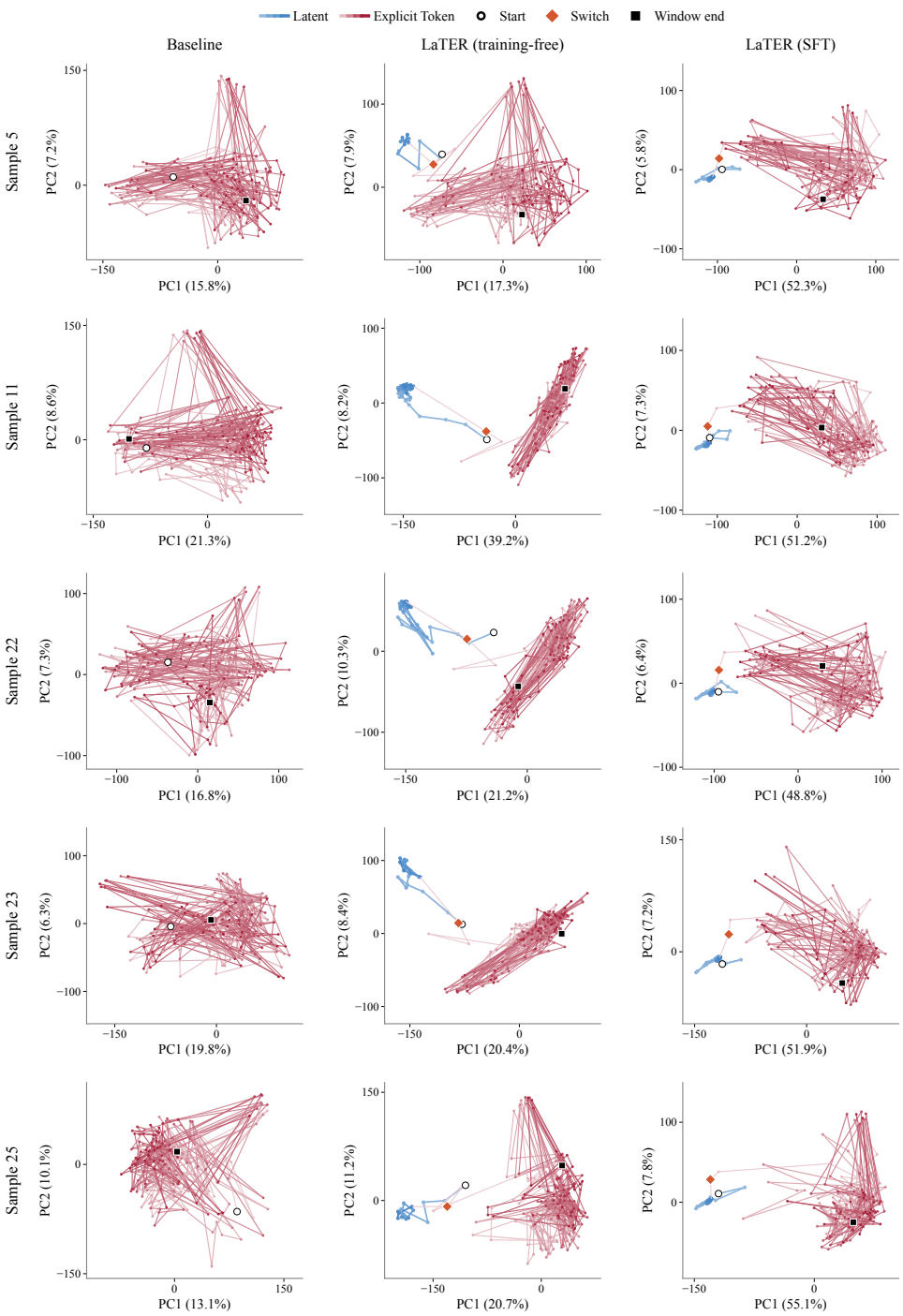

Figure 12: **Qwen3-14B on AIME 2025: samples 5, 11, 22, 23, and 25.** Columns show CoT, LaTER-TF, and LaTER-SFT. Colors and markers follow Figure 9.

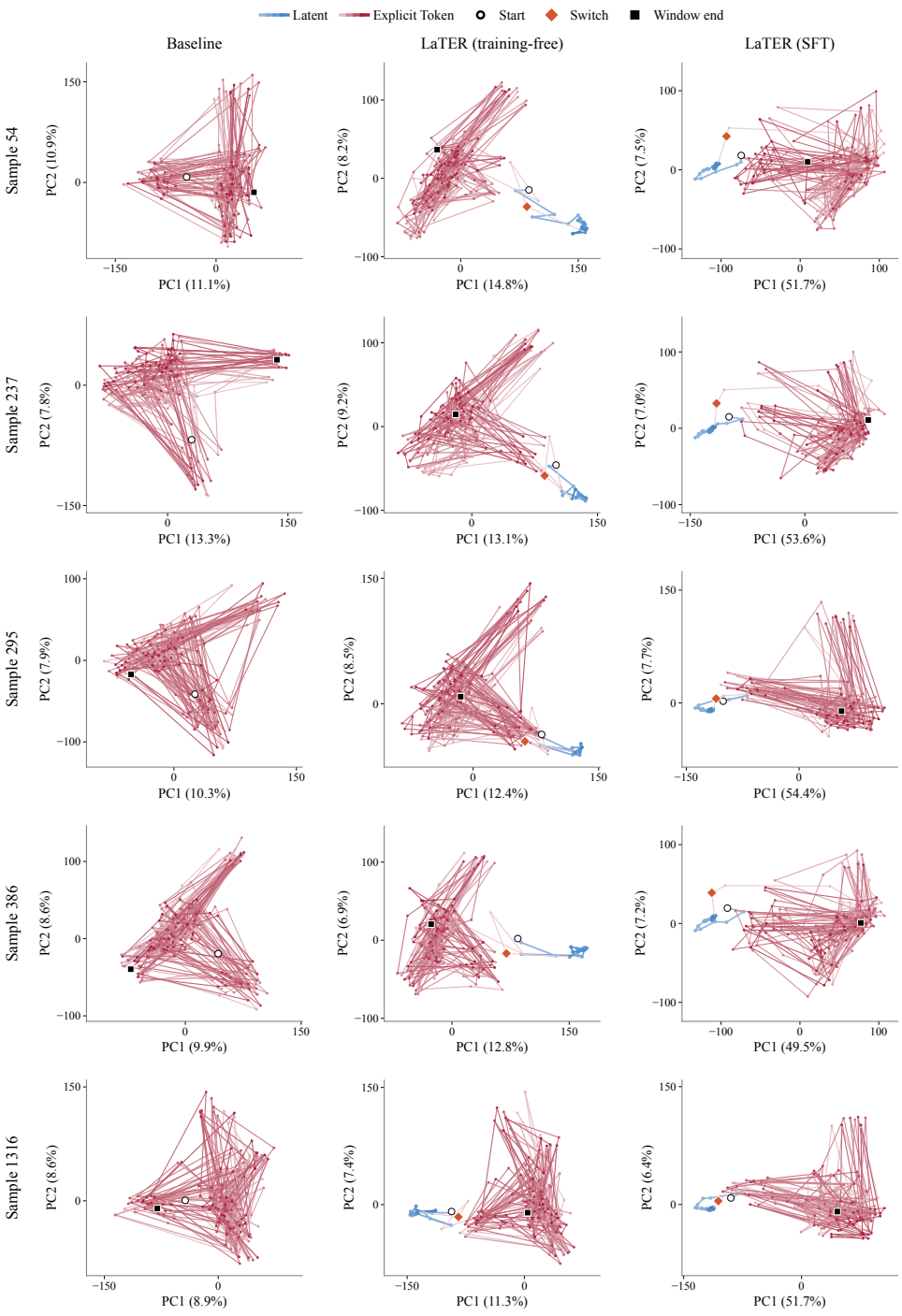

Figure 13: **Qwen3-14B on GSM8K: samples 54, 237, 295, 386, and 1316.** Columns show CoT, LaTER-TF, and LaTER-SFT.

D PROMPTS

The templates below reproduce the evaluation and data-construction prompts, including their placeholders and delimiters. The first three specify evaluation answer formats; the remaining four form the system/user pairs used to extract intuitions and generate continuations.

D.1 EVALUATION PROMPTS

Prompt

```
You are a helpful assistant.  Given an input question, think about
what methods should be used and corresponding detailed step-by-step
plan to solve the question.

Question:  {question}

Your outlined plan should be concise with a few bulletpoints for
each step.  During planning stage, only describe the methods and
steps; do not carry out the full solution or produce the final
answer.

In the solution stage, output the final answer inside
\boxed{YOUR_FINAL_ANSWER}.
Now output your plan to solve the question below:
```

Figure 14: Prompt for math questions.

Prompt

```
You are a helpful assistant.  Given an input question, think about
what methods should be used and corresponding detailed step-by-step
plan to solve the question.

Question:  {question}

Your outlined plan should be concise with a few bulletpoints for
each step.  During planning stage, only describe the methods and
steps; do not carry out the full solution or produce the final
answer.

In the solution stage, output the correct option letter inside
\boxed{YOUR_FINAL_ANSWER}.
Now output your plan to solve the question below:
```

Figure 15: Prompt for multi-choice questions.

Prompt

````
You are a helpful assistant.  Given an input question, think about
what methods should be used and corresponding detailed step-by-step
plan to solve the question.

Target Question:  {question}

You must put all python code as self-contained Python function(s)
in markdown code blocks.  For example:

```python
import math
def add(a, b):
````
````

```
  return a + b
```

Your outlined plan should be concise with a few bulletpoints for each step. During planning stage, only describe the methods and steps; do not carry out the full solution or produce the final answer.
In the solution stage, provide the complete working Python implementation in a Python code block.
Now output your plan to solve the question below:

Figure 16: Prompt for code problems.

D.2 Data-construction prompts

Prompt

```
You are an expert reasoning data curator.

Extract only the key insights from the source reasoning.

Rules:
- Return valid JSON only.
- Do not produce a short chain of thought.
- Do not provide any final answers in your response.
- correct_insight must be the coarse but correct high-level solution idea.
- incorrect_insights must only include wrong ideas that are explicitly evidenced in the source outputs.
- Do not invent errors that are not present in the source outputs.
```

Figure 17: System prompt to generate solution intuition.

Prompt

```
You will receive a problem, one or more source outputs, and optional ground truth.

Return JSON with this schema:
{
  ''task_summary'': ''short task type description'',
  ''correct_insight'': ''2--10 sentences of high-level correct plan'',
  ''incorrect_insights'': [
    {
      ''idea'': ''2--10 sentences of a wrong high-level plan evidenced in the source outputs'',
      ''why_wrong'': ''brief reason it fails''
    }
  ],
  ''source_answer_correct'': true,
  ''contains_reflection'': true
}

REMEMBER: Only extract insights that reflect on the high-level idea. **Do not** give any actual answer in correct_insight or incorrect_insights.

Only give your complete plan to solve the question. Do not directly state whether it is right or wrong. Do not add content
```

```
unrelated to the idea in correct_insight or incorrect_insights.

Problem prompt:
<<<PROMPT>>>
{prompt}
<<<END_PROMPT>>>

Source outputs:
<<<OUTPUTS>>>
{outputs}
<<<END_OUTPUTS>>>

Ground truth:
<<<GROUND_TRUTH>>>
{ground_truth}
<<<END_GROUND_TRUTH>>>

Metadata:
{metadata_json}
```

Figure 18: User prompt to generate solution intuition.

Prompt

```
You are solving a problem from your previous intuition.

Rules:
- Continue from your intuition instead of restarting from scratch.
- Your intuition may be correct or incorrect.
- Follow your intuition and finish the reasoning efficiently.
- Keep private reasoning compact and avoid repeated planning.
- In the visible answer, provide the final answer and a brief
justification only.
```

Figure 19: System prompt to generate shorter CoT.

Prompt

```
Solve the following problem by continuing from your intuition.

Your intuition may be correct or incorrect.  Do not ignore it.
Continue from it and finish the solution.

REMEMBER: The intuition comes from your previous conversation with
yourself.  It is not the user's intuition.

Problem prompt:
<<<PROMPT>>>
{prompt}
<<<END_PROMPT>>>

Your Intuition:
<<<INTUITION>>>
{insight_text}
<<<END_INTUITION>>>
```

Figure 20: User prompt to generate shorter CoT.

E External Assets

Table 8 identifies the external models, source data, benchmarks, and implementation libraries.

Table 8: External assets and their published license information.

Asset / source	Role	Published terms
Qwen3-8B, Qwen3-14B (Yang et al., 2025)	Inference and supervised backbones	Apache-2.0
DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025)	Training-free backbone	MIT; underlying Llama 3.1 terms also identified by the model card
OLMo-3-32B-Think (Olmo et al., 2026)	Training-free backbone	Apache-2.0
Dolci-Think-SFT-32B (Olmo et al., 2026)	Source reasoning traces	ODC-BY; component-specific terms also apply
MathArena (Balunović et al., 2025)	AIME 2025 evaluation	MIT for repository materials; original question rights are separate
MATH-500 (Lightman et al., 2023)	Mathematics evaluation	No license specified on the cited dataset card
GSM8K (Cobbe et al., 2021)	Mathematics evaluation	MIT
GPQA Diamond (Rein et al., 2024)	Science evaluation	MIT
ARC-Challenge (Clark et al., 2018)	Reasoning evaluation	CC-BY-SA-4.0 on the dataset card
HumanEval+, MBPP+ (Liu et al., 2023)	Code evaluation	Apache-2.0 for EvalPlus; MIT also applies to its HumanEval-derived evaluation code
FlashAttention-2 (Dao, 2023)	Attention implementation	BSD-3-Clause
DeepSpeed / ZeRO-3 (Rajbhandari et al., 2020)	Distributed training	Apache-2.0