REVISITING LATENT CHAIN-OF-THOUGHT: DISENTANGLING REPRESENTATION AND SUPERVISION*

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent chain-of-thought (CoT) reasoning aims to accelerate inference by replacing explicit text tokens with continuous hidden states. Existing approaches implement this concept through diverse latent representations and training objectives, but the tight coupling of these elements obscures the impact of individual design choices. To address this, we propose **LatentGrid**, a unified framework that decouples representation constraints from supervision strategies. By systematically varying representation constraints and supervision strategies under a shared experimental setup, we isolate their individual and joint effects on task accuracy, rationale recoverability, distributional alignment, and optimization dynamics. Our analysis indicates that unconstrained continuous representations paired with direct rationale-derived supervision better preserve recoverable reasoning information, while curriculum learning facilitates the transition to implicit reasoning. Conversely, restricting latent states to vocabulary embedding mixtures generally degrades performance and limits representational capacity. Inspired by these findings, we further propose **A**daptive **C**urriculum **T**hinking (ACT), a novel training paradigm that integrates curriculum learning with direct and token-level supervision. This synthesis improves reasoning accuracy and enables adaptive generation lengths, suggesting that effective latent reasoning benefits from flexible representations, explicit intermediate targets, and a progressive shift from explicit text.

1 INTRODUCTION

Chain-of-thought (CoT) prompting significantly improves the accuracy of large language models (LLMs) on complex tasks by generating intermediate reasoning steps prior to the final answer (Wei et al., 2022). However, because LLMs generate text autoregressively, the lengthy reasoning traces produced by CoT inevitably increase inference cost and latency (Hao et al., 2024; Shen et al., 2025; Tan et al., 2026). Latent chain-of-thought (latent CoT) methods address this bottleneck by shifting intermediate computations from discrete natural-language tokens into continuous hidden states. In this way, these methods aim to preserve the advantages of CoT reasoning while reducing the substantial overhead introduced by autoregressive generation (Hao et al., 2024; Shen et al., 2025).

Current latent CoT methods implement continuous computation through diverse designs, differing primarily in how they represent and supervise latent states. On the representation front, methods like Coconut recurrently feed raw hidden states back into the model, whereas Soft Thinking and Latent-SFT constrain states to weighted mixtures of vocabulary embeddings (Hao et al., 2024; Zhang et al., 2026; Deng et al., 2025). On the supervision front, training objectives range from indirect downstream optimization (CODI) to direct alignment with explicit rationales, such as step reconstruction (SIM-CoT) and compressed embedding prediction (CoLaR) (Shen et al., 2025; Wei et al., 2026; Tan et al., 2026). While recent studies have begun probing the mechanics of latent reasoning (Zhu et al., 2026; Liang & Pan, 2026; Chen et al., 2026), their analyses remain largely confined to a specific method or a single design dimension. Because existing works often tightly bundle specific representation choices with particular supervision objectives under disparate experimental setups, direct comparisons remain difficult. Consequently, a fundamental question remains open: **How do these distinct choices of representation and supervision actually interact to shape a model's reasoning capabilities and internal dynamics?**

*Our code is available at `https://anonymous.4open.science/r/LatentGrid-7253`

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.

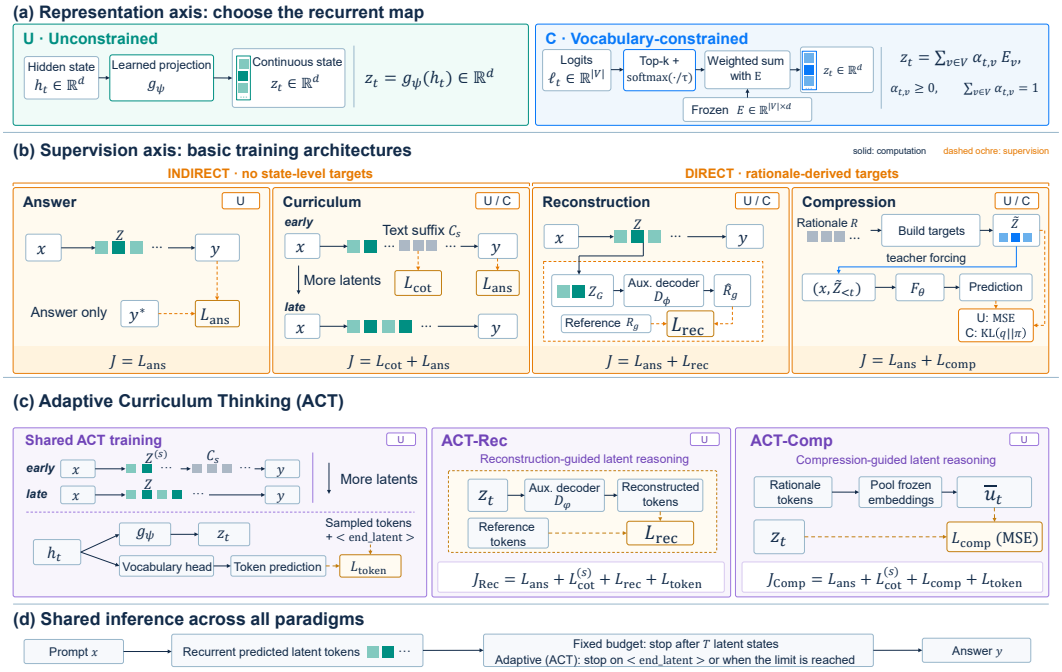


Figure 1: Overview of LatentGrid and ACT. (a) The representation axis specifies the recurrent map: unconstrained continuous states or vocabulary-constrained embedding mixtures. (b) The supervision axis distinguishes indirect optimization from direct rationale-derived targets. (c) ACT progressively internalizes explicit reasoning with token-level grounding, using reconstruction-guided (ACT-Rec) or compression-guided (ACT-Comp) latent states. (d) All paradigms share the same latent recurrence and answer-generation procedure, with either a fixed budget or adaptive stopping.

To answer this question, we introduce **LatentGrid**, a structured framework that systematically decouples latent representation from supervision strategies, enabling a controlled and rigorous investigation. Specifically, LatentGrid organizes latent CoT along two orthogonal axes: *the representation axis*, which specifies whether latent states are modeled as unconstrained continuous vectors or restricted to mixtures of vocabulary embeddings, and *the supervision axis*, which determines whether training proceeds indirectly via downstream objectives or directly through explicit intermediate reasoning targets. Under a shared experimental protocol, we instantiate representative paradigms across this grid to examine both downstream task accuracy and internal mechanics through rationale recoverability probes, distributional alignment metrics, and training gradient dynamics. Informed by our diagnostic findings, we introduce **A**daptive **C**urriculum **T**hinking (ACT), a unified training paradigm that progressively shifts computation to continuous states while bridging the discrete-continuous modality gap through token-level grounding. We instantiate this paradigm through two concrete variants: **ACT-Rec**, which guides continuous trajectories via step reconstruction, and **ACT-Comp**, which supervises them through rationale compression. By easing the structural shift from explicit text and eliminating the constraint of fixed-budget reasoning, ACT enables models to dynamically allocate latent steps based on problem difficulty while substantially improving reasoning accuracy. Overall, our controlled analysis identifies three design principles for effective latent reasoning: (i) preserving representational flexibility, (ii) providing explicit intermediate supervision, and (iii) progressively transitioning from textual to latent reasoning. We instantiate these principles in ACT, which further introduces token-level grounding and adaptive termination.

Our main contributions are threefold: **1)** We establish LatentGrid, a structured framework that systematically decouples latent CoT along two orthogonal axes—representation constraints and supervision sources—providing a principled foundation to isolate and investigate core design choices under a controlled setting. **2)** Across this grid, we conduct an extensive empirical study encompassing task accuracy, rationale recoverability, distributional alignment, and gradient tracking, revealing how representation constraints and supervision signals independently and jointly govern implicit reasoning mechanics. **3)** Informed by these diagnostic insights, we introduce ACT, instantiated via

ACT-Rec and ACT-Comp. By uniting progressive transition with token-level alignment, ACT substantially improves reasoning accuracy while enabling adaptive, difficulty-aware trajectory lengths.

2 Related Work

Latent chain-of-thought methods. Latent CoT methods replace explicit rationale tokens with continuous hidden states to condition subsequent computations. Existing approaches differ primarily in how these states are represented and supervised. In terms of representation, methods like Coconut and CoLaR utilize unconstrained continuous vectors derived from hidden states or learned mappings (Hao et al., 2024; Tan et al., 2026). Conversely, Soft Thinking and Latent-SFT constrain these states to weighted mixtures of vocabulary embeddings (Zhang et al., 2026; Deng et al., 2025). On the supervision front, methods differ in the source of their training signals. Approaches like Coconut and CODI rely entirely on indirect signals, guiding the latent trajectory through downstream prediction losses or teacher activation matching (Hao et al., 2024; Shen et al., 2025). Other methods provide direct supervision by anchoring latent states to targets derived from explicit rationales. These targets take diverse forms, including selected hidden states in CCoT, reconstruction objectives in SIM-CoT, compressed embeddings in CoLaR, and learned vocabulary-space representations in Latent-SFT (Cheng & Van Durme, 2024; Wei et al., 2026; Tan et al., 2026; Deng et al., 2025). However, because these choices are typically coupled within disparate model architectures and evaluation settings, isolating the impact of any single design element remains challenging.

Mechanistic studies of latent reasoning. Recent studies have investigated the interpretability of latent states and their underlying reasoning mechanics. For example, Zhu et al. (2026) analyze reasoning dynamics in Coconut, demonstrating that continuous states can maintain multiple reasoning paths in graph reachability tasks. Other efforts investigate interpretability by decoding explicit reasoning traces from latent states (Dilgren & Wiegreffe, 2026), or analyze causal structures via interventions on latent steps (Li et al., 2026). From a training perspective, Chen et al. (2026) evaluate various supervision strategies and their effects on the information preserved in latent states. While these studies provide detailed insights into latent reasoning, their analyses remain focused on particular methods or research directions. To provide a broader perspective, we decouple representation and supervision within a unified framework. By systematically evaluating their individual and joint effects under a shared protocol, we clarify how these dimensions shape latent reasoning and leverage these insights to synthesize more effective training strategies.

3 LatentGrid: Decoupling Representation and Supervision

We formalize LatentGrid by structuring latent CoT along two orthogonal axes. Specifically, *the representation axis* specifies whether latent states are modeled as unconstrained continuous vectors or restricted to mixtures of vocabulary embeddings, while *the supervision axis* determines whether training proceeds indirectly via downstream objectives or directly through explicit intermediate reasoning targets. Figure 1 illustrates this framework, depicting the shared recurrence process at inference alongside the representative design choices. Appendix A provides a schematic overview of the paradigms and representative methods along both axes. In this section, we formalize the implicit reasoning process (Section 3.1) and detail the implementation of these representation constraints (Section 3.2) and supervision strategies (Section 3.3).

3.1 Implicit Reasoning Process

Implicit reasoning is typically formulated as a recursive process over continuous states, thereby eliminating the need to instantiate each reasoning step as explicit text. Let x denote the input, $Z = (z_1, \ldots, z_T)$ represent the latent trajectory, and $y = (y_1, \ldots, y_{|y|})$ denote the final answer. Starting from the prompt $P(x)$, the language model F_θ iteratively generates:

$$(h_t, \ell_t) = F_\theta(P(x), z_{<t}), \qquad z_t = \rho(h_t, \ell_t), \qquad t = 1, \ldots, T, \tag{1}$$

where h_t is the final-layer hidden state, ℓ_t represents the vocabulary logits, and ρ denotes a mapping function that derives the next continuous input from the model's current output. At each step, z_t is appended to the context window prior to the next recurrence. After completing T steps, the model exits the latent mode and autoregressively generates the textual response y based on $P(x)$ and Z.

3.2 LATENT REPRESENTATIONS

Existing methods typically structure the latent space in one of two ways: as unconstrained continuous vectors or as constrained mixtures of vocabulary embeddings. To evaluate this axis, we instantiate one representative mapping for each. Throughout all implementations, the pretrained token embeddings remain frozen.

Unconstrained representations. Unconstrained latent states are modeled as continuous vectors $z_t \in \mathbb{R}^d$ without explicit vocabulary constraints, as seen in methods like Coconut and CoLaR (Hao et al., 2024; Tan et al., 2026). While this provides representational flexibility, the lack of explicit vocabulary constraints allows the vectors to drift from the pretrained representation manifold, thereby introducing distributional misalignment (Deng et al., 2025; Chen et al., 2026). In our framework, we instantiate these states using a learned nonlinear projection of the current hidden state:

$$z_t^{\text{unc}} = g_\psi(h_t), \tag{2}$$

where $g_\psi : \mathbb{R}^d \to \mathbb{R}^d$ is a trainable mapping.

Constrained representations. Conversely, constrained methods restrict each latent state to a structured geometric region, typically the convex hull of the pretrained vocabulary embeddings (e.g., Soft Thinking, Latent-SFT (Zhang et al., 2026; Deng et al., 2025)). Confining latent vectors to this hull regularizes their geometry and may reduce representation drift, though overly rigid constraints risk bottlenecking representational capacity (Chen et al., 2026). We operationalize this constraint by selecting a top-k support set $\mathcal{S}_t$ from the original vocabulary logits. We normalize these logits using temperature τ to form a convex combination of the token embeddings E_v:

$$\alpha_{t,\mathcal{S}_t} = \text{softmax}(\ell_{t,\mathcal{S}_t}/\tau)\,, \qquad z_t^{\text{con}} = \sum_{v \in \mathcal{S}_t} \alpha_{t,v} E_v. \tag{3}$$

3.3 SUPERVISION STRATEGIES

The second major design axis dictates how the latent states are trained: either indirectly through downstream prediction tasks or directly via intermediate signals derived from explicit rationales.

3.3.1 INDIRECT SUPERVISION

Indirect supervision optimizes the latent trajectory without assigning predefined targets to individual latent states. The recurrent generation process is guided purely by downstream objectives.

Answer supervision. In its simplest form, the model generates a fixed-length latent trajectory Z and is supervised exclusively through the final-answer loss:

$$\mathcal{L}_{\text{ans}} = -\log p_\theta(y \mid x, Z)\,. \tag{4}$$

This provides no explicit state-level guidance about what each latent state should encode.

Curriculum supervision. To provide denser guidance, curriculum supervision progressively replaces explicit reasoning steps with latent states (Hao et al., 2024). At training stage s, the first s explicit reasoning steps are replaced by a latent prefix $Z^{(s)}$, while the remaining explicit rationale is retained as a textual suffix C_s. The stage-wise objective combines standard language modeling on the suffix with the final-answer prediction:

$$\mathcal{J}_{\text{cur}}^{(s)} = \mathcal{L}_{\text{cot}}^{(s)} + \mathcal{L}_{\text{ans}}^{(s)}. \tag{5}$$

As training advances, the latent prefix expands and the explicit suffix contracts. Since the latent states are still generated freely without predefined targets, the supervision remains indirect.

3.3.2 DIRECT SUPERVISION

In contrast, direct supervision explicitly constructs intermediate targets from an available explicit rationale $R = (r_1, \ldots, r_K)$ to ground the latent computations. Final-answer prediction remains the shared downstream objective, such that the complete direct objective is written as $\mathcal{J}_{\text{dir}} = \mathcal{L}_{\text{ans}} + \mathcal{L}_{\text{dir}}$, where $\mathcal{L}_{\text{dir}}$ represents one of the following auxiliary losses.

Reconstruction supervision. This approach ensures that latent states retain sufficient information to recover specific explicit reasoning steps, as explored in SIM-CoT (Wei et al., 2026). We operationalize this by partitioning the latent trajectory into groups $Z_1, \ldots, Z_G$ and pairing them with explicit rationale segments $R_1, \ldots, R_G$. An auxiliary decoder reconstructs each segment from its associated latent group. Writing N_R for the total number of valid target tokens, the reconstruction loss is:

$$\mathcal{L}_{\text{rec}} = -\frac{1}{N_R} \sum_{g=1}^{G} \sum_{j=1}^{|R_g|} \log p_\phi(r_{g,j} \mid Z_g, r_{g,<j}) \,. \tag{6}$$

The auxiliary decoder provides gradients for the latent states during training but is discarded at inference.

Compression supervision. Compression supervision constructs compact continuous targets from rationale tokens and aligns the predicted trajectory accordingly (e.g., CoLaR (Tan et al., 2026)). We implement this by partitioning the tokenized explicit rationale into consecutive, non-overlapping blocks B_i, each containing up to κ tokens. For each block, we construct a single target embedding u_i by pooling the pretrained embeddings of its constituent tokens:

$$u_i = \frac{1}{\sqrt{|B_i|}} \sum_{c \in B_i} E_c, \tag{7}$$

where the factor $1/\sqrt{|B_i|}$ stabilizes the scale. The training objective $\mathcal{L}_{\text{comp}}$ then depends on the chosen representation constraint:

Unconstrained compression: The pooled vector directly serves as the target. We normalize u_i by the population standard deviation of the embedding coordinates (s_E) and train the latent head via mean squared error (MSE):

$$\bar{u}_i = \frac{u_i}{s_E}, \qquad \mathcal{L}_{\text{comp}}^{\text{unc}} = \frac{1}{M} \sum_{i=1}^{M} \text{MSE}(g_\psi(h_i), \bar{u}_i) \,. \tag{8}$$

The prediction is then rescaled by s_E before recurrence.

Constrained compression: The pooled vector u_i defines a sparse teacher distribution over the vocabulary. We score the vocabulary embeddings via dot product $a_{i,v} = u_i^\top E_v$, retain the top-k support $\mathcal{S}_i$, and apply a softmax to obtain the teacher distribution $q_i(v)$. This teacher defines the target state $\widetilde{z}_i^{\text{con}} = \sum_{v \in \mathcal{S}_i} q_i(v) E_v$ and supervises the predicted vocabulary distribution $\pi_{\theta,i}$ via KL divergence:

$$\mathcal{L}_{\text{comp}}^{\text{con}} = \frac{1}{M} \sum_{i=1}^{M} D_{\text{KL}}(q_i \,\|\, \pi_{\theta,i}) \,. \tag{9}$$

During training, these rationale-derived targets are teacher-forced as recurrent inputs. At inference, the model generates the trajectory autoregressively through its respective representation map.

4 Empirical Evaluation across LatentGrid

We instantiate the representative paradigms across LatentGrid under a shared experimental protocol to investigate how representation and supervision influence latent reasoning. This setup enables a direct comparison of their effects on both downstream task performance and internal reasoning mechanics.

4.1 Experimental Setup

Data and Training Setup. Our primary experiments use Llama-3.2-1B-Instruct as the backbone and GSM8K-Aug (Shen et al., 2025) for training. All latent variants and the No-CoT baseline are initialized from a shared checkpoint fine-tuned on explicit CoT, which also serves as the explicit-CoT baseline. We evaluate in-domain accuracy on GSM8K (Cobbe et al., 2021) and generalization on SVAMP (Patel et al., 2021), GSM-Hard (Gao et al., 2023), and MultiArith (Roy & Roth, 2015). Full training and inference configurations are detailed in Appendix B.1. To further assess generalizability across model families and non-mathematical tasks, we also report experiments using Qwen3-0.6B on mathematical benchmarks and test ProsQA in Appendix D.

Table 1: Answer accuracy (%) and rationale reconstruction for all latent variants. Latent models generate a fixed budget of 12 latent states, while explicit CoT averages 25.62, 11.98, 32.93, and 14.24 reasoning tokens across the respective tasks. BERTScore and answer consistency are evaluated on the 1,301 GSM8K test examples with explicit rationales. Bold and underlined values indicate the best and second-best latent results per column.

		Answer accuracy ↑				Rationale reconstruction		
Objective	Rep.	GSM8K	SVAMP	GSM-Hard	MultiArith	NLL ↓	BERT ↑	Cons. ↑
No-CoT	–	32.83	44.40	7.35	62.33	–	–	–
Explicit CoT	–	60.50	64.20	14.63	96.83	–	–	–
Answer	U	31.69	42.60	7.05	61.67	0.287	0.9036	16.30
Curriculum	U	**35.03**	**47.70**	7.96	67.83	0.395	0.8930	15.30
Curriculum	C	30.63	<u>46.40</u>	6.90	59.17	0.768	0.8417	0.85
Reconstruction	U	<u>34.50</u>	45.30	<u>8.19</u>	66.83	**0.195**	**0.9319**	31.44
Reconstruction	C	24.64	43.80	5.53	52.67	0.803	0.8456	0.69
Compression	U	33.36	39.20	**9.17**	**80.83**	<u>0.217</u>	<u>0.9293</u>	<u>40.66</u>
Compression	C	34.27	35.00	<u>8.19</u>	<u>69.17</u>	0.347	0.9285	**41.12**

Compared Paradigms. Our evaluation covers four supervision paradigms: indirect methods (Answer and Curriculum) and direct methods (Reconstruction and Compression). While Answer uses only unconstrained states, the remaining three are evaluated under both unconstrained and constrained settings. During evaluation, each latent variant recursively generates 12 latent states before producing the final answer. In addition, we benchmark two standard baselines: No-CoT, which outputs answers directly, and explicit CoT, which produces a full textual rationale prior to answering.

Task and Probe Evaluation. We assess reasoning performance through final-answer accuracy across all benchmarks. In addition, we evaluate the rationale content retained in latent states by training an auxiliary probe for each frozen reasoner to reconstruct explicit rationales from its full trajectory. All probes share an identical architecture and training protocol. We report validation reconstruction loss (NLL), test BERTScore (BERT), and answer consistency (Cons.) to quantify how effectively reasoning steps can be recovered. Appendix B.2 provides full details on probe implementation and interpretation.

4.2 Main Results

Table 1 reports both the downstream answer accuracy and the explicit-rationale reconstruction performance for the seven latent CoT variants alongside the reference baselines.

Under unconstrained representations, curriculum learning and direct supervision generally improve task accuracy. Both Curriculum and Reconstruction outperform Answer and No-CoT across all four benchmarks, raising GSM8K accuracy from 31.69% (Answer) to 35.03% and 34.50%, respectively. Compression also exceeds both baselines on three benchmarks, reaching 80.83% on MultiArith compared to 61.67% for Answer.

Probe evaluations indicate that direct supervision preserves more recoverable reasoning content. With unconstrained states, Reconstruction and Compression outperform Answer and Curriculum across all three probe metrics. In particular, Reconstruction lowers validation NLL to 0.195 (versus 0.287 for Answer), while Compression achieves 40.66% answer consistency compared to 16.30% for Answer and 15.30% for Curriculum.

Conversely, restricting latent states to vocabulary embedding mixtures generally degrades rationale recovery. For all three applicable paradigms, constrained states result in higher probe NLL and lower BERTScore. This effect is most pronounced in Reconstruction, where NLL climbs from 0.195 to 0.803 and BERTScore falls from 0.9319 to 0.8456.

4.3 Analysis and Discussion

Direct Supervision and Information Recoverability. The advantage of direct supervision in rationale recovery stems from how its training objectives structure intermediate states. Unlike indirect

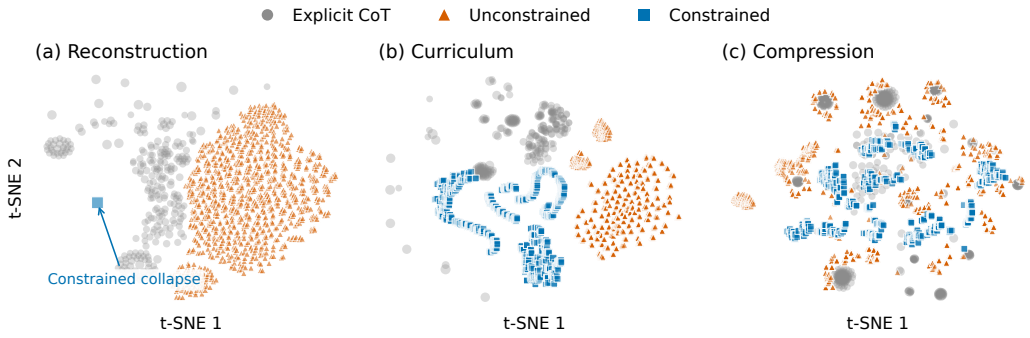


Figure 2: t-SNE projections of explicit CoT embeddings and U/C latent states for Reconstruction, Curriculum, and Compression. The sampled Constrained Reconstruction states are identical before projection. Both Compression variants overlap more closely with explicit CoT.

methods that guide latent trajectories solely through downstream losses, direct supervision explicitly defines which information should be preserved in latent states. As detailed in Appendix B.3, requiring latent groups to reproduce explicit reasoning segments directly optimizes a variational lower bound on the mutual information between latent states and the reference text. Similarly, Compression guides continuous states to match pooled representations of explicit steps. By anchoring latent trajectories to concrete reasoning content, both approaches better preserve task-relevant information during recurrent unrolling, providing a more reliable foundation for downstream answer prediction.

Representation Constraints, Alignment, and Information Content. While vocabulary constraints are designed to prevent representation drift, our earlier results indicate that they lower task accuracy and degrade rationale recovery. To understand this outcome, we evaluate whether these constraints actually achieve distributional alignment with explicit CoT embeddings. As shown by the t-SNE projections in Figure 2 and the squared maximum mean discrepancy (MMD^2) in Table 2 (Gretton et al., 2012), restricting continuous states to vocabulary mixtures does not reliably align them with the explicit CoT embedding manifold. Constrained Reconstruction collapses to a single point, and Constrained Curriculum retains a noticeable discrepancy. In contrast, Compression maintains close alignment with explicit CoT representations under both unconstrained and constrained settings.

This pattern stems from the difference between imposing geometric boundaries and providing distribution-level guidance. Restricting states to the convex hull limits their spatial bounds but cannot regulate their empirical distribution across inputs. In contrast, Compression integrates alignment directly into the optimization objective. By supervising continuous states with pooled rationale embeddings and preserving their variance through square-root scaling, it naturally anchors unconstrained states to the explicit CoT manifold. Appendix B.4 elaborates on why geometric constraints alone cannot guarantee alignment and how the compression objective facilitates it.

Table 2: MMD^2 between latent states and explicit CoT representations, reported with 95% question-level bootstrap confidence intervals. Lower values indicate closer alignment.

Objective	Unconstrained	Constrained
Reconstruction	0.405 $[0.400, 0.410]$	0.532 $[0.528, 0.536]$
Curriculum	0.494 $[0.491, 0.497]$	0.221 $[0.213, 0.229]$
Compression	0.046 $[0.042, 0.050]$	0.072 $[0.068, 0.076]$

Beyond distributional shift, a more fundamental concern is whether geometric constraints restrict the amount of information continuous states can represent. As reflected by our probe evaluations across all three supervision objectives, constrained states lead to poorer rationale recovery. This loss of recoverable content suggests that forcing continuous states into mixtures of vocabulary embeddings impairs their capacity to distinguish between fine-grained reasoning steps. Consequently, our findings highlight the benefit of preserving representational flexibility while relying on targeted supervision to steer distributional alignment.

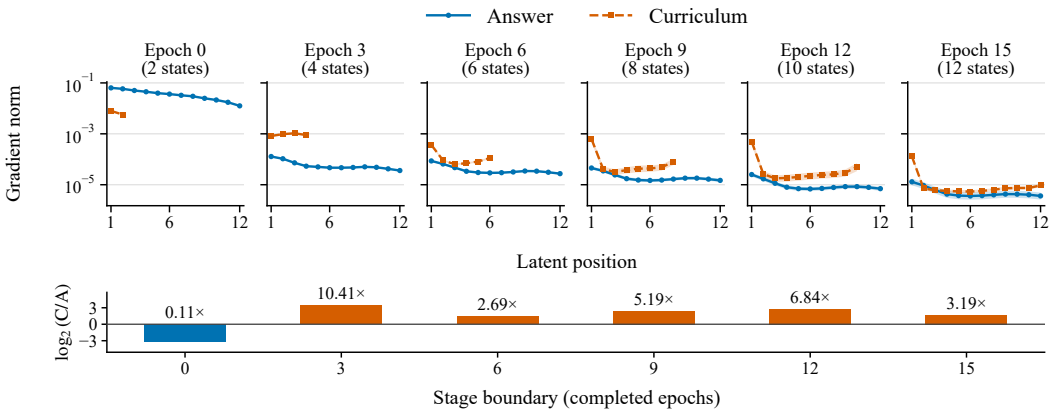


Figure 3: Latent-state gradient norms for Unconstrained Answer and Curriculum across curriculum stage boundaries. Top: mean gradient norm at each latent position, reflecting the magnitude of the training signal received by individual states. Bottom: ratio of Curriculum to Answer mean norms over their shared latent positions ($\log_2$ scale), comparing the overall strength of the learning signal between the two training objectives. Measurement details are in Appendix B.5.

Curriculum Learning and Optimization Dynamics. The performance advantage of Curriculum over Answer highlights the critical role of optimization dynamics in learning latent reasoning. By progressively substituting early reasoning steps with continuous states while retaining explicit text for subsequent steps, curriculum training provides dense intermediate signals that guide the latent prefix. To quantify these training signals, we record local gradient norms at each latent position over the first five optimizer updates across stage transitions.

As shown in Figure 3, except for the initial stage, Curriculum produces larger gradient norms across latent positions at every stage boundary. This stronger optimization signal persists even into the final stage where explicit rationale text is entirely removed. These results suggest that curriculum learning provides stronger intermediate optimization signals during the transition to continuous reasoning, which may contribute to its improved downstream performance. Appendix B.5 provides full details on this experiment.

5 From Diagnostic Insights to Adaptive Curriculum Thinking

While our LatentGrid analysis demonstrates the benefits of unconstrained representations and direct supervision, standalone latent paradigms still lag behind explicit CoT in downstream accuracy. We trace this gap to two key bottlenecks: (1) optimization instability caused by the abrupt transition from discrete text to continuous recurrence, and (2) the semantic disconnect between continuous latents and the pretrained discrete vocabulary, which is compounded by rigid, fixed-length generation budgets. To address these limitations, we propose **A**daptive **C**urriculum **T**hinking (ACT), a unified training paradigm that couples progressive curriculum learning with discrete token grounding, as illustrated in Figure 1(c). We instantiate this paradigm through two concrete variants: **ACT-Rec** (paired with reconstruction targets) and **ACT-Comp** (paired with compression targets). This design substantially improves reasoning accuracy while enabling difficulty-aware adaptive trajectory lengths. Table 3 summarizes the empirical results across all variants, with detailed configurations and loss formulations provided in Appendix C.1.

Progressive integration for direct targets. We begin by addressing the optimization challenge of transitioning to continuous reasoning. While direct targets explicitly define the information latent states should capture, abruptly shifting from discrete text to continuous recurrence often destabilizes training. To provide a smoother structural transition, we adopt a curriculum learning approach, progressively expanding a latent prefix and substituting explicit text in stages. As shown in the top section of Table 3, this progressive integration improves accuracy over the standalone direct supervision evaluated earlier. For instance, Progressive Reconstruction improves GSM8K accuracy from

Table 3: Performance of intermediate training paradigms leading to ACT. Each entry reports answer accuracy (%) followed by the mean number of generated latent states in parentheses. Variants without adaptive stopping use a fixed budget of 12 states. † marks variants that adjust the reconstruction target window to align with the token-level grouping.

Training paradigm	GSM8K	SVAMP	GSM-Hard	MultiArith
Direct supervision with curriculum learning				
Progressive Reconstruction	45.41 (12.00)	<u>57.10</u> (12.00)	10.92 (12.00)	**95.00** (12.00)
Progressive Compression	34.34 (12.00)	41.00 (12.00)	8.79 (12.00)	79.00 (12.00)
Direct supervision with token-level grounding				
Token-grounded Reconstruction†	36.16 (12.93)	49.90 (6.03)	8.34 (15.43)	68.50 (7.39)
Token-grounded Compression	40.49 (13.03)	46.40 (6.03)	8.79 (15.24)	78.50 (7.30)
Adaptive Curriculum Thinking				
ACT-Rec†	**49.51** (12.77)	**58.90** (5.98)	<u>10.99</u> (15.11)	<u>93.50</u> (7.23)
ACT-Comp	<u>49.13</u> (12.91)	55.90 (6.10)	**11.52** (16.75)	93.33 (7.31)

34.50% to 45.41%. This suggests that explicit intermediate targets are internalized more effectively when the model is given time to gradually adapt to continuous computation.

Bridging the modality gap with token-level grounding. Beyond the structural transition, another fundamental discrepancy remains: latent states operate in a continuous space, whereas the backbone model was pretrained exclusively on discrete text. To bridge this modality gap, we introduce an auxiliary token-level cross-entropy (CE) objective. Rather than relying solely on continuous target matching, this objective explicitly trains the model to predict corresponding discrete reasoning tokens directly from the intermediate latent states, thereby grounding the continuous trajectory in the model's native vocabulary distribution. Furthermore, this grounding naturally resolves the limitation of fixed-budget reasoning—by training the model to predict a termination token at the end of the trajectory, it acquires the ability to halt generation adaptively. As shown in the middle section of Table 3, introducing this token-level grounding improves downstream accuracy. The learned stopping behavior also dynamically scales with problem difficulty, allocating fewer states to simpler tasks (averaging 6.0–7.4 states on SVAMP and MultiArith) while extending the reasoning trajectory for harder problems like GSM-Hard.

The ACT paradigm. Integrating these structural and semantic solutions yields our complete training paradigm, ACT. As demonstrated in Table 3, the two ACT variants yield the strongest overall performance among the evaluated training paradigms across the benchmarks. In particular, ACT-Rec reaches 49.51% accuracy on GSM8K, improving over basic Reconstruction by 15.01 percentage points, and reaches 58.90% on SVAMP, while ACT-Comp achieves 11.52% on GSM-Hard. Ultimately, these results suggest that effective latent reasoning benefits from anchoring continuous states to the discrete vocabulary while progressively shifting computation away from explicit text.

6 Conclusion

Latent CoT reasoning reduces LLM inference latency by shifting discrete steps into continuous hidden states. However, the tight coupling of design choices in existing methods obscures the impact of individual components. To address this, we introduced LatentGrid, a unified framework to systematically evaluate latent CoT by decoupling representation constraints from supervision strategies. Our findings reveal that unconstrained representations, when paired with direct rationale-derived supervision and curriculum learning, better preserve reasoning content and ease optimization. Conversely, forcing latent states into vocabulary mixtures restricts representational capacity without reliably resolving distributional shift. Guided by these insights, we proposed Adaptive Curriculum Thinking. By integrating progressive curriculum schedules, direct intermediate targets, and token-level grounding, ACT substantially improves reasoning accuracy while enabling models to adaptively adjust their latent trajectory lengths. Despite these advancements, computational constraints limited our experiments to smaller models. Furthermore, while ACT substantially narrows the performance gap, latent reasoning still trails explicit CoT. Closing this accuracy gap and extending our findings to larger models and broader domains remain key directions for future research.

AI USE STATEMENT

In this work, we used generative AI tools to design or provide feedback on research methodology or experiments (specifically, suggesting the MMD metric for distributional alignment) and to assist in the writing of proofs (optimizing mathematical formulations and derivations). We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, implement methods, or interpret results. Furthermore, tasks such as generating synthetic data sets, assisting with translation, cleaning and reformatting datasets, and supporting qualitative and thematic data analysis are not applicable to this work.

Additionally, we used generative AI tools for several recommended tasks: to identify relevant literature, create or edit software code (including assisting in the generation of algorithmic pseudocode), create or modify scientific figures (specifically, assisting in generating the conceptual framework diagrams and optimizing the visual readability of Figure 3), draft parts of the research paper, and edit the manuscript to improve readability.

We have reviewed all AI-assisted work. Specifically, literature identified by AI was manually cross-referenced on Google Scholar to verify relevance and prevent hallucinated citations. All AI-assisted mathematical formulations and derivations were verified for theoretical soundness, and the generated algorithmic pseudocode and analysis code were manually reviewed and tested for correctness. We take full responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Xinghao Chen, Chak Tou Leong, Wenjin Guo, Jian Wang, Wenjie Li, and Xiaoyu Shen. What makes effective supervision in latent chain-of-thought: An information-theoretic analysis. *arXiv preprint arXiv:2606.20075*, 2026.

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

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.

Jingcheng Deng, Liang Pang, Zihao Wei, Shicheng Xu, Zenghao Duan, Kun Xu, Yang Song, Huawei Shen, and Xueqi Cheng. Llm latent reasoning as chain of superposition. *arXiv preprint arXiv:2510.15522*, 2025.

Connor Dilgren and Sarah Wiegreffe. Are latent reasoning models easily interpretable? *arXiv preprint arXiv:2604.04902*, 2026.

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.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The journal of machine learning research*, 13:723–773, 2012.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Zirui Li, Xuefeng Bai, Kehai Chen, Yizhi Li, Jian Yang, Chenghua Lin, and Min Zhang. Dynamics within latent chain-of-thought: An empirical study of causal structure. *arXiv preprint arXiv:2602.08783*, 2026.

Jia Liang and Liangming Pan. Do latent-cot models think step-by-step? a mechanistic study on sequential reasoning tasks. *arXiv preprint arXiv:2602.00449*, 2026.

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, 2021.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4609–4622, 2020.

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, 2015.

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

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of llm reasoning chains. *Advances in Neural Information Processing Systems*, 38:4646–4668, 2026.

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.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

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.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart J Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. *Advances in Neural Information Processing Systems*, 38:79931–79963, 2026.

A Overview of LatentGrid

LatentGrid provides a unified framework by decoupling how latent states are represented from how they are supervised. Figure 4 summarizes the two axes introduced in Section 3, illustrating the evaluated training paradigms alongside representative existing methods.

B Supplementary Details for the Controlled Evaluation

This appendix provides the experimental protocols and supporting analyses for Section 4.

B.1 Experimental Setup

B.1.1 Datasets

We train on GSM8K-Aug (Shen et al., 2025), an augmented version of the GSM8K training set containing approximately 385K examples. Each example pairs a question with a numeric answer and an explicit rationale expressed as a sequence of calculations. GSM8K (Cobbe et al., 2021) provides the in-domain evaluation set of 1,319 grade-school word problems that require several arithmetic steps.

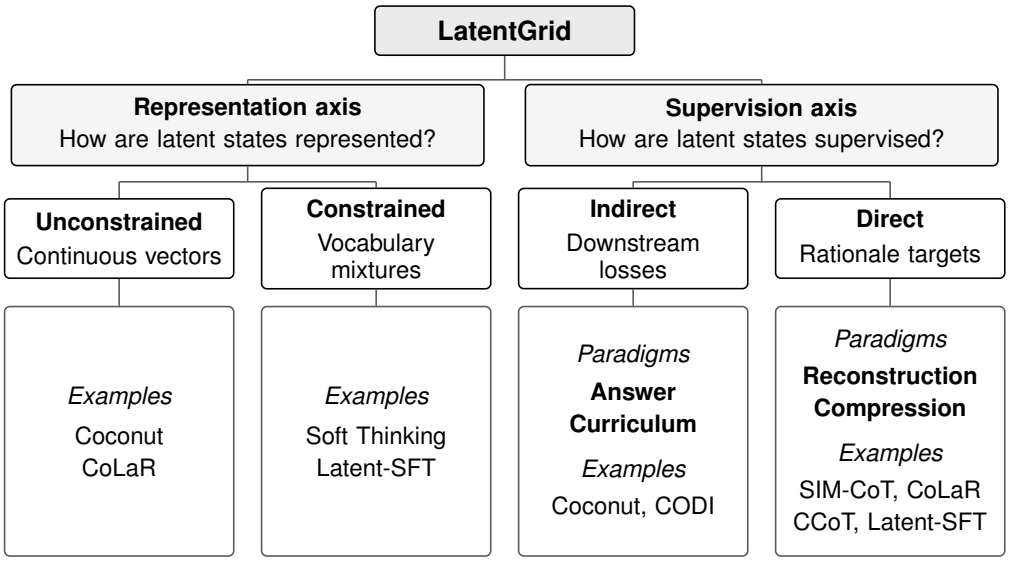


Figure 4: Overview of LatentGrid. The representation axis distinguishes unconstrained continuous states from vocabulary-constrained mixtures. The supervision axis groups Answer and Curriculum under indirect supervision, and Reconstruction and Compression under direct supervision. Representative existing methods are categorized accordingly.

Three further benchmarks assess generalization. SVAMP (Patel et al., 2021) contains 1,000 arithmetic word problems created by varying the wording and structure of existing problems. GSM-Hard (Gao et al., 2023) modifies the 1,319 GSM8K test questions by replacing a number in each question with a larger value, testing arithmetic with larger operands. MultiArith (Roy & Roth, 2015) contains 600 short word problems that combine multiple arithmetic operations. Table 4 gives an example from each dataset.

Table 4: Example questions and reference answers from GSM8K-Aug training data and the four evaluation datasets. The paired GSM8K and GSM-Hard examples illustrate the change in numerical scale.

Dataset	Question	Answer
GSM8K-Aug	A farmer raises 15 chickens. Each chicken is capable of laying 5 eggs in a week. How many eggs will the farmer collect at the end of 3 weeks?	225
GSM8K	A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?	3
SVAMP	Each pack of dvds costs 76 dollars. If there is a discount of 25 dollars on each pack. How much do you have to pay to buy each pack?	51
GSM-Hard	A robe takes 2287720 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?	3431580
MultiArith	Bianca had 45 coloring books. If she gave away 6 of them, but then bought 20 more, how many would she have total?	59

Reference CoT and Reasoning Steps. Each <<...>> block represents one reasoning step in the reference CoT. For the GSM8K-Aug question in Table 4, the reference CoT is

```
<<5*3=15>>
<<15*15=225>>
```

The first step computes the number of eggs produced by one chicken over three weeks. The second multiplies this quantity by 15 chickens to obtain the total of 225 eggs.

B.1.2 Training and Inference

Shared setup. All latent models and the No-CoT baseline use Llama-3.2-1B-Instruct, initialized from the same explicit-CoT checkpoint. This checkpoint also provides the explicit-CoT baseline. We train each latent model and No-CoT for 25 epochs using AdamW, with an effective batch size of 256 and a learning rate of 2×10^{-4}. The learning rate follows a cosine schedule with 3% warm-up; weight decay is 0.01, and training uses bfloat16.

We fine-tune the models using LoRA with rank 128, alpha 32, and dropout 0.1. Unconstrained states are produced by a two-layer projection with hidden width 512, GELU activation, and layer normalization. Constrained states use convex combinations of the top 10 vocabulary embeddings at temperature 1. The embedding table remains frozen.

Answer and Curriculum. Answer recursively generates 12 latent states, feeding each state back into the model before predicting the next. The resulting trajectory conditions final-answer prediction, and training minimizes cross-entropy for the final answer. No-CoT uses the same answer objective but predicts directly from the question.

Curriculum gradually replaces the initial `<<...>>` reasoning steps with a recursively generated latent prefix. During epochs 1–15, successive three-epoch stages use 2, 4, 6, 8, and 10 states, replacing the first one to five steps, respectively. From epoch 16 onward, the model uses 12 states and removes all remaining rationale text. The retained rationale and final answer are learned through teacher forcing, with their separately token-averaged cross-entropy losses added together. Model parameters carry over between stages, while the AdamW moment estimates are reset at each transition.

Reconstruction. The reasoner recursively generates 12 states and learns to predict the final answer from this trajectory. For direct supervision, the states are divided into six consecutive pairs. The first five pairs reconstruct the first five rationale steps, and the sixth reconstructs all remaining steps. If a corresponding step is absent, that pair instead reconstructs the final answer.

An auxiliary Llama-3.2-1B-Instruct decoder with a separate LoRA adapter reconstructs each segment from its associated pair of states. The decoder is initialized from the base model and trained jointly with the reasoner. Reconstruction loss is averaged over all valid target tokens and added to the answer loss with equal weight. Its gradients propagate through the latent states to the reasoner.

Compression. We compress every two consecutive CoT tokens into one latent token by pooling their embeddings. Unconstrained Compression fits these vectors using mean squared error after dividing them by s_E, the population standard deviation of vocabulary embedding coordinates. The unnormalized pooled vectors are used as recurrent inputs. Constrained Compression instead derives a sparse teacher distribution over 10 vocabulary embeddings from each pooled vector and learns it through KL divergence. Its recurrent input is the corresponding weighted sum of vocabulary embeddings.

During training, each target state is fed back as the input for the following step. The final answer is also conditioned on this target trajectory. Training minimizes the sum of the latent prediction loss and answer cross-entropy. Thus, the training trajectory length follows the rationale length, while evaluation uses a fixed 12 states.

Inference. Inference begins by encoding the question followed by `<start_latent>`. The current hidden state is mapped to a latent vector, which becomes the input for the next step. In Unconstrained Compression, the predicted vector is multiplied by s_E before being fed back. All seven latent variants repeat this process for exactly 12 states. The model then appends `<end_latent>` and the answer cue `###Answer:`, and generates answer tokens autoregressively using greedy decoding. Reconstruction follows the same procedure with its auxiliary decoder removed.

B.2 Probe Configuration and Interpretation

Probe analysis examines the rationale information recoverable from each method's latent states and how this relates to task performance.

B.2.1 PROBE CONFIGURATION

We collect each method's complete 12-state latent trajectories and train a separate probe to reconstruct the corresponding reference rationales from these latents alone.

All probes use Llama-3.2-1B-Instruct with LoRA rank 128, alpha 32, and dropout 0.1. We train on GSM8K-Aug for 25 epochs with an effective batch size of 256 and a learning rate of 2×10^{-4}. The learning rate follows a cosine schedule with 3% warm-up; weight decay is 0.01, and training uses bfloat16 and seed 42. For each probe, we retain the checkpoint with the lowest validation loss for evaluation.

B.2.2 PROBE METRICS

Table 1 summarizes probe performance using the following three metrics.

Validation reconstruction loss. We report validation negative log-likelihood (NLL) per target token, with lower values indicating better reconstruction of the reference rationales.

Test BERTScore. On the GSM8K test set, mean BERTScore F1 (Zhang et al., 2019) measures semantic similarity between generated and reference rationales; higher scores indicate greater similarity. We use layer 9 of `bert-base-uncased`, without IDF weighting or baseline rescaling.

Test answer consistency. On the GSM8K test set, answer consistency is the percentage of examples where the reconstructed numeric answer matches the corresponding model prediction. Missing answers count as disagreements.

B.2.3 INFORMATION-THEORETIC INTERPRETATION OF THE PROBE

Reconstruction loss and rationale information. From an information-theoretic perspective, minimizing the probe's reconstruction loss is equivalent to maximizing a variational lower bound on the mutual information between the latent trajectories and the reference rationales. Let R denote the complete reference rationale, Z the latent trajectory, and $q_\phi(R \mid Z)$ the probe's conditional distribution. Under the joint data distribution $p(R, Z)$, adding and subtracting $\log p(R \mid Z)$ expands the sequence reconstruction loss as

$$\begin{aligned}\mathcal{L}_{\text{probe}}(\phi) &= \mathbb{E}_{p(R,Z)}[-\log q_\phi(R \mid Z)] \\ &= \mathbb{E}_{p(R,Z)}\left[-\log p(R \mid Z) + \log \frac{p(R \mid Z)}{q_\phi(R \mid Z)}\right] \\ &= H(R \mid Z) + \mathbb{E}_{p(Z)} D_{\text{KL}}(p(R \mid Z) \,\|\, q_\phi(R \mid Z))\,.\end{aligned} \tag{10}$$

Here, $H(R \mid Z)$ measures the uncertainty about the reference rationale after observing the latent trajectory. Since the KL term is non-negative,

$$H(R \mid Z) \le \mathcal{L}_{\text{probe}}(\phi). \tag{11}$$

Substituting this inequality into the definition of mutual information gives the variational bound (Pimentel et al., 2020):

$$\begin{aligned}I(R; Z) &= H(R) - H(R \mid Z) \\ &\ge H(R) - \mathcal{L}_{\text{probe}}(\phi).\end{aligned} \tag{12}$$

Consequently, optimizing the probe to minimize reconstruction loss directly maximizes this lower bound. The probe loss thus serves as a practical measure of how much rationale information is preserved and recoverable from the latent representations.

Connection to answer prediction. Clearer and more easily recoverable rationale information generally facilitates more accurate answer generation. We can formalize this relationship by considering an answer decoder $d_\omega(Y \mid Q, R)$ that predicts the correct answer Y based on the question Q and the rationale R. Combining it with the probe defines a hypothetical composed answer distribution:

$$\begin{aligned}\widetilde{p}_{\phi,\omega}(Y \mid Q, Z) &= \sum_r q_\phi(r \mid Z) d_\omega(Y \mid Q, r) \\ &\ge q_\phi(R \mid Z) d_\omega(Y \mid Q, R).\end{aligned} \tag{13}$$

The inequality follows by retaining the term for the reference rationale R. Taking the negative logarithm of both sides yields an upper bound on the loss for a single example:

$$-\log \widetilde{p}_{\phi,\omega}(Y \mid Q, Z) \leq -\log q_\phi(R \mid Z) - \log d_\omega(Y \mid Q, R). \tag{14}$$

Taking expectations over $p(Q, R, Y, Z)$ then yields

$$\begin{aligned} \mathcal{L}_{\text{composed}} &= \mathbb{E}[-\log \widetilde{p}_{\phi,\omega}(Y \mid Q, Z)] \\ &\leq \mathbb{E}[-\log q_\phi(R \mid Z)] + \mathbb{E}[-\log d_\omega(Y \mid Q, R)] \\ &= \mathcal{L}_{\text{probe}} + \mathcal{L}_{\text{answer|rationale}}. \end{aligned} \tag{15}$$

Here, $\mathcal{L}_{\text{answer|rationale}}$ is the expected answer loss when the reference rationale is provided. For a fixed answer decoder and data distribution, this term is constant. Lower reconstruction loss therefore reduces the upper bound on the composed decoder's answer loss.

We acknowledge a discrepancy between this theoretical construction and the actual implicit reasoning process, which generates answers directly from the latent trajectory without explicitly reconstructing the rationale. Nevertheless, this bound illustrates that preserving recoverable reasoning-basis information in the latent space is beneficial for downstream answer generation.

B.3 Information-Theoretic Interpretation of Reconstruction Supervision

Let R_g denote a reference rationale segment and Z_g its corresponding latent group. Their joint distribution $p_\theta(R_g, Z_g)$ is induced by the training data and the latent generator. For the auxiliary decoder $p_\phi(R_g \mid Z_g)$, the expected sequence reconstruction loss decomposes as

$$\begin{aligned} \ell_g(\theta, \phi) &= -\mathbb{E}_{p_\theta(R_g, Z_g)} \log p_\phi(R_g \mid Z_g) \\ &= H_\theta(R_g \mid Z_g) + \mathbb{E}_{p_\theta(Z_g)} D_{\mathrm{KL}}(p_\theta(R_g \mid Z_g) \,\|\, p_\phi(R_g \mid Z_g)) \\ &\geq H_\theta(R_g \mid Z_g). \end{aligned} \tag{16}$$

The final inequality follows from the non-negativity of KL divergence. Substituting it into the definition of mutual information gives the variational bound (Pimentel et al., 2020):

$$\begin{aligned} I_\theta(R_g; Z_g) &= H(R_g) - H_\theta(R_g \mid Z_g) \\ &\geq H(R_g) - \ell_g(\theta, \phi). \end{aligned} \tag{17}$$

With the data distribution and segmentation fixed, $H(R_g)$ is independent of the model parameters. Minimizing ℓ_g therefore maximizes this variational lower bound, offering an information-theoretic interpretation of how reconstruction supervision encourages each latent group to retain decodable rationale information.

B.4 Representation Constraints and Distributional Alignment

B.4.1 Distributional Comparison

For each of Curriculum, Reconstruction, and Compression, we collect 12 latent states from the unconstrained and constrained models on the same 128 validation questions. Token embeddings generated by the shared explicit-CoT checkpoint provide the reference distribution. For each objective, we sample 1,000 points from the explicit, unconstrained, and constrained representations and fit a joint t-SNE projection. We use perplexity 30, PCA initialization, 1,000 iterations, and seed 42.

For the quantitative comparison, we use the complete matched trajectories after ℓ_2 normalization. We compute MMD^2 for each question and obtain 95% confidence intervals by bootstrapping across questions.

Maximum mean discrepancy. Let $Z = \{z_i\}_{i=1}^n$ and $X = \{x_j\}_{j=1}^m$ denote the latent states and matched explicit CoT embeddings for one question. We compute the biased empirical squared maximum mean discrepancy:

$$\widehat{\mathrm{MMD}}_b^2(Z, X) = \frac{1}{n^2} \sum_{i,i'} k(z_i, z_{i'}) + \frac{1}{m^2} \sum_{j,j'} k(x_j, x_{j'}) - \frac{2}{nm} \sum_{i,j} k(z_i, x_j). \tag{18}$$

We use an equally weighted multi-bandwidth RBF kernel,

$$k(a,b) = \frac{1}{L}\sum_{\ell=1}^{L} \exp\left(-\frac{\|a-b\|_2^2}{2\sigma_\ell^2}\right), \tag{19}$$

where the bandwidths $\{\sigma_\ell\}_{\ell=1}^{L}$ are selected from the pooled representations using the median heuristic. We compute $\widehat{\mathrm{MMD}}_b^2$ for each question and average the resulting values across questions. Lower values indicate greater similarity between the latent and explicit CoT representation distributions.

B.4.2 Why the Convex-Hull Constraint Does Not Ensure Alignment

Let $C \sim p_{\mathrm{CoT}}$ denote an explicit CoT token and let $X = E_C$ be its embedding. The explicit representation distribution is

$$P_X = \sum_{v\in\mathcal{V}} p_{\mathrm{CoT}}(v)\delta_{E_v}. \tag{20}$$

A constrained latent state is defined as

$$Z = \sum_{v\in\mathcal{V}} \alpha_v E_v, \qquad \alpha \in \Delta^{|\mathcal{V}|-1}. \tag{21}$$

The distribution of Z depends on how the learned coefficients α vary across inputs. Its first and second moments are

$$\begin{aligned} \mathbb{E}[Z] &= \sum_v \mathbb{E}[\alpha_v]E_v, \\ \mathbb{E}[ZZ^\top] &= \sum_{v,w} \mathbb{E}[\alpha_v\alpha_w]E_vE_w^\top. \end{aligned} \tag{22}$$

For the explicit token distribution P_X, the corresponding moments are

$$\begin{aligned} \mathbb{E}[X] &= \sum_v p_{\mathrm{CoT}}(v)E_v, \\ \mathbb{E}[XX^\top] &= \sum_v p_{\mathrm{CoT}}(v)E_vE_v^\top. \end{aligned} \tag{23}$$

Equality of the distributions requires equality of these moments. However, the simplex constraint only enforces non-negative coefficients summing to one for each latent state. It does not require their statistics across inputs to reproduce the moments of P_X. Consequently, convex hull membership restricts the possible locations of latent states but does not guarantee that their distribution matches that of explicit CoT embeddings.

B.4.3 How Compression Promotes Distributional Alignment

Compression constructs latent targets directly from explicit token embeddings using square-root scaling. Let $X_1, \ldots, X_m$ denote the embeddings of explicit tokens in a rationale block, with common mean μ and covariance Σ. The unconstrained target is

$$U = \frac{1}{\sqrt{m}}\sum_{j=1}^{m} X_j. \tag{24}$$

Its mean and covariance are

$$\mathbb{E}[U] = \sqrt{m}\mu, \qquad \mathrm{Cov}(U) = \Sigma + \frac{1}{m}\sum_{j\neq l}\mathrm{Cov}(X_j, X_l). \tag{25}$$

When the embedding distribution is approximately centered and the aggregate correlation within each block is small,

$$\mathbb{E}[U] \approx 0, \qquad \mathrm{Cov}(U) \approx \Sigma. \tag{26}$$

For independent embeddings drawn from $\mathcal{N}(0,\Sigma)$, the pooled target follows the same distribution:

$$U \sim \mathcal{N}(0,\Sigma). \tag{27}$$

By comparison, ordinary averaging gives covariance Σ/m under independence. Square-root scaling therefore preserves the original variance, ensuring the latent targets maintain a scale consistent with the explicit token embeddings.

Constrained Compression, however, further maps these pooled targets to sparse convex combinations of vocabulary embeddings. For $m > 1$, square-root scaling gives coefficients summing to $\sqrt{m} > 1$, so the pooling operation is not a convex combination. As a result, the generated targets may lie outside the vocabulary convex hull, and projecting them back can alter their distribution. This may explain the greater distributional discrepancy of Constrained Compression relative to Unconstrained Compression in Table 2.

B.5 Latent-Position Gradient Analysis

Training and measurement windows. We train the Unconstrained Answer and Curriculum models on GSM8K-Aug using the shared explicit-CoT initialization. Both models follow the configurations in Appendix B.1 and process training examples in the exact same order. To monitor training dynamics, we record gradients during the first five optimizer updates immediately following epochs 0, 3, 6, 9, 12, and 15. At these stage boundaries, the Curriculum model progressively increases its latent states from 2 to 12 in increments of two, while the Answer model uses a fixed budget of 12 states throughout. Training proceeds normally between these measurement windows and concludes after the fifth update of epoch 16, by which point the Curriculum model has completely phased out the explicit rationale text.

Gradient measurement. For model M at stage boundary b, we extract the gradient for every recurrent latent state $z_{i,t}$ of each training example i during the standard backward pass. Let $\mathcal{J}$ denote the training objective. As defined in Section 3.3, the Answer model is optimized solely via the final-answer loss ($\mathcal{J} = \mathcal{L}_{\text{ans}}$). The Curriculum model optimizes the stage-wise sum of the textual rationale loss and the final-answer loss ($\mathcal{J} = \mathcal{J}_{\text{cur}}^{(s)} = \mathcal{L}_{\text{cot}}^{(s)} + \mathcal{L}_{\text{ans}}^{(s)}$). We compute the Euclidean norm of these gradients across the latent dimensions, preserving the standard loss normalization and backward scaling:

$$G_{i,t}^{M,b} = \left\| \frac{\partial \mathcal{J}}{\partial z_{i,t}} \right\|_2 . \tag{28}$$

Aggregation and confidence intervals. At the beginning of each curriculum stage, we collect gradients across five optimizer updates. With an effective batch size of 256, this yields $N = 1{,}280$ gradient observations per latent position. We average these observations to obtain the mean gradient norm at each position, $\bar{G}_t^{M,b} = N^{-1} \sum_{i=1}^{N} G_{i,t}^{M,b}$. To establish confidence intervals, we apply bootstrap resampling. We resample the example indices with replacement 1,000 times, maintaining the exact pairing across models and latent positions. The 2.5th and 97.5th percentiles of these bootstrapped means form the pointwise 95% confidence intervals.

Plot construction and shared-position comparison. Figure 3 illustrates these dynamics. The six upper panels display the position-wise mean gradient norms and their confidence bands on a common logarithmic vertical axis. Each panel title indicates the number of completed epochs and the number of latent states used by Curriculum at that stage.

The lower panel compares the relative magnitude of the training signals. Let m_b denote the number of latent states used by Curriculum at stage boundary b. We compute the ratio R_b over the first m_b latent positions shared by both models:

$$R_b = \frac{m_b^{-1} \sum_{t=1}^{m_b} \bar{G}_t^{\text{Curriculum},b}}{m_b^{-1} \sum_{t=1}^{m_b} \bar{G}_t^{\text{Answer},b}}. \tag{29}$$

The bar chart plots the base-2 logarithm of this ratio ($\log_2 R_b$), and the text annotations display the raw ratio values.

Table 5: Configurations for ACT and its precursor variants, all using unconstrained representations. † denotes Reconstruction with one state per group of up to two CoT tokens.

Training paradigm	Curriculum	Token CE	Inference states
Progressive rationale supervision			
Progressive Reconstruction	Yes	No	Fixed 12
Progressive Compression	Yes	No	Fixed 12
Token-grounded reasoning			
Token-grounded Reconstruction†	No	Yes	Adaptive
Token-grounded Compression	No	Yes	Adaptive
Adaptive Curriculum Thinking			
ACT-Rec†	Yes	Yes	Adaptive
ACT-Comp	Yes	Yes	Adaptive

C Supplementary Details for ACT and Its Precursor Variants

This appendix details the training objectives, token targets, and latent-budget evaluation for ACT and the precursor variants in Section 5.

C.1 Supervision Design and Experimental Setup

C.1.1 Token-Level CE Loss

Following Compression, we divide the tokenized reference CoT into consecutive groups of two explicit tokens, with one latent token corresponding to each group. If the token count is odd, the final group contains one token. We use a token sampled uniformly from each group as the CE target at the position generating its corresponding latent token. The vocabulary predictor includes the latent boundary tokens and uses the same hidden state as the latent-vector predictor. The CE loss averages the negative log-likelihood of these sampled tokens and one `<end_latent>` target per example over all valid target positions in a minibatch. In curriculum stages with a retained rationale, the end-marker target marks the transition from the latent prefix to the textual suffix.

C.1.2 Experimental Configurations

Adaptive inference requires at least one latent state and stops when the highest-scoring vocabulary token is `<end_latent>`, subject to a limit of 64 states. Answer generation is capped at 64 tokens. Reported latent lengths exclude boundary markers and answer tokens. Table 5 summarizes the six configurations.

Progressive Rationale Supervision. Both variants use the reference CoT steps defined in Appendix B.1.1 as curriculum units. Progressive Reconstruction attaches the original reconstruction target to each pair of latent states introduced by the curriculum. Progressive Compression matches the pooled embedding sequence to these step boundaries as closely as the two-token grouping allows. A token pair can cross a step boundary; we keep it intact and assign it to the later step. The embeddings assigned to the steps covered by the current stage form the supervised latent prefix. Direct supervision continues at the final stage without a retained rationale.

Token-Grounded Reasoning. Token-grounded Compression adds token CE to its existing training objective. Token-grounded Reconstruction adopts the same correspondence as Compression: one latent token corresponds to two consecutive explicit CoT tokens. This replaces the basic Reconstruction assignment of two latent states to one reasoning segment. The auxiliary decoder reconstructs both explicit tokens and its end-of-sequence marker, while CE supervises the reasoner's vocabulary prediction with one sampled token from the same pair. A final one-token group is handled in the same way. Both variants supervise all rationale token groups from the start.

Adaptive Curriculum Thinking. Both ACT variants use one latent token for every two explicit tokens. We therefore match their latent prefixes to the reference reasoning steps using the same

complete token pairs as Progressive Compression. Step boundaries are followed as closely as possible without splitting a pair, and a pair crossing a boundary is assigned to the later step. Direct supervision and token CE apply to the selected latent prefix, while cross-entropy on the remaining explicit tokens is retained during intermediate stages.

D Additional Evaluations on Qwen3-0.6B and ProsQA

To test the generalizability of our findings, we extend our empirical evaluation along two complementary dimensions: model architecture and reasoning domain. First, to assess cross-model consistency, we evaluate the seven basic latent paradigms alongside ACT-Rec and ACT-Comp on Qwen3-0.6B across the four arithmetic benchmarks. Second, to verify generalization beyond arithmetic, we evaluate the two ACT variants on ProsQA (Hao et al., 2024), a non-mathematical logical reasoning benchmark.

D.1 Experimental Setup

Datasets and tasks. For the arithmetic evaluation, we follow the same setup described in Appendix B, training Qwen models on GSM8K-Aug and evaluating on GSM8K, SVAMP, GSM-Hard, and MultiArith. For non-mathematical reasoning, we employ ProsQA (Hao et al., 2024), where each problem requires navigating a directed acyclic graph (DAG) of logical dependencies formulated in natural language. We evaluate Llama-3.2-1B-Instruct on the 500 test instances of ProsQA.

Implementation details. Training and probe protocols match those described in Appendices B and C. For Qwen3-0.6B, the seven basic latent variants generate a fixed budget of 12 latent states. For the token-grouped variants, we adjust the grouping ratio to achieve comparable trajectory budgets across models: each latent state corresponds to three explicit tokens for Qwen on arithmetic tasks, and four explicit tokens for Llama on ProsQA. As in the primary experiments, the two ACT variants predict a termination signal and halt adaptively.

For reference, explicit-CoT baselines average 36.20, 15.68, 58.31, and 17.80 reasoning tokens on GSM8K, SVAMP, GSM-Hard, and MultiArith for Qwen3-0.6B, and 35.96 reasoning tokens on ProsQA for Llama-3.2-1B-Instruct.

D.2 Cross-Architecture Evaluation: Qwen3-0.6B

Tables 6 and 7 report downstream task accuracy and probe reconstruction metrics for Qwen3-0.6B, respectively.

Consistency with primary findings. The empirical results on Qwen3-0.6B closely mirror our main findings on Llama-3.2-1B-Instruct. First, curriculum supervision proves essential for stabilizing implicit reasoning: while pure answer supervision collapses to 10.92% on GSM8K, curriculum training provides consistent intermediate signals that raise accuracy to 37.38%. Second, direct supervision remains superior at preserving recoverable reasoning traces. As Table 7 shows, unconstrained Reconstruction and Compression outperform indirect approaches across all probe metrics, achieving substantially lower validation NLL and higher answer consistency. Finally, vocabulary constraints consistently degrade downstream accuracy and rationale recovery across all paradigms, reinforcing our finding that restricting continuous states to vocabulary embedding mixtures limits representational capacity.

Effectiveness of ACT. ACT also improves reasoning accuracy on Qwen3-0.6B, extending the gains from progressive, token-grounded training to this backbone. ACT-Rec achieves the highest overall accuracy across all four datasets, outperforming the best basic variant by 9.02% on GSM8K (reaching 46.40%) and 17.83% on MultiArith (reaching 94.00%). Furthermore, the adaptive stopping mechanism exhibits the same desirable task-difficulty scaling: the models dynamically allocate fewer states on simpler benchmarks (roughly 5.2 to 6.3 states on SVAMP and MultiArith) while naturally expanding their reasoning trajectories on GSM-Hard (exceeding 16 states).

Table 6: Answer accuracy (%) for Qwen3-0.6B. U and C denote unconstrained and constrained representations. The seven basic variants use 12 latent states; parentheses in the final two rows indicate the mean adaptive trajectory length. Bold and underlined values mark the best and second-best latent accuracy per column.

		Answer accuracy ↑			
Objective	Rep.	GSM8K	SVAMP	GSM-Hard	MultiArith
No-CoT	–	35.78	51.60	8.49	75.50
Explicit CoT	–	65.43	73.70	30.63	96.17
Answer	U	10.92	39.90	3.18	15.00
Curriculum	U	37.38	53.60	<u>9.48</u>	76.17
Curriculum	C	34.19	52.30	8.79	73.33
Reconstruction	U	30.86	50.20	7.20	70.83
Reconstruction	C	11.90	42.40	3.41	15.33
Compression	U	33.28	42.90	8.57	72.50
Compression	C	6.14	15.80	1.90	10.67
ACT-Rec	U	**46.40** (11.56)	**61.10** (5.19)	**11.14** (16.47)	**94.00** (6.24)
ACT-Comp	U	<u>40.41</u> (12.03)	<u>54.00</u> (5.68)	9.33 (18.60)	<u>87.67</u> (6.28)

Table 7: Rationale reconstruction for the seven basic Qwen3-0.6B variants on GSM8K. Evaluation metrics follow Table 1: validation NLL, test BERTScore F1, and test answer consistency (%). Bold and underlined values mark the best and second-best results per column.

Objective	Rep.	NLL ↓	BERT ↑	Cons. ↑
Answer	U	0.251	0.8892	4.69
Curriculum	U	0.229	0.8984	16.14
Curriculum	C	0.343	0.8734	6.07
Reconstruction	U	<u>0.219</u>	0.9027	<u>19.52</u>
Reconstruction	C	0.312	0.8776	3.54
Compression	U	**0.169**	**0.9282**	**36.13**
Compression	C	0.225	<u>0.9121</u>	3.77

D.3 Cross-Domain Evaluation: Llama-3.2-1B-Instruct on ProsQA

To test whether our conclusions transfer to non-mathematical domains, we evaluate ACT-Rec and ACT-Comp on ProsQA. As reported in Table 8, latent reasoning exhibits a clear advantage in this logical reasoning setting. Both ACT variants achieve 98.20% accuracy, surpassing explicit CoT (96.60%). More importantly, they achieve this higher accuracy with a dramatic reduction in sequence length, requiring only 8.56 and 8.69 continuous states on average compared to 35.96 tokens for explicit CoT—a more than 75% reduction in reasoning steps. This result highlights that when equipped with proper supervision and structural grounding, continuous latent states can perform multi-step relational deductions more compactly and accurately than explicit text generation.

E Pseudocode for Inference and Training

This section provides the pseudocode for the shared inference procedure, the seven basic LatentGrid training paradigms, the two ACT variants, and probe training.

Data format and templates. Each training example consists of a question x, a reference rationale R, and a final answer y. We format the model input using the following template:

Question: x Let's think step by step:###
<start_latent>Z<end_latent>C###Answer:$y\|$EOS

Table 8: Performance of Llama-3.2-1B-Instruct on the 500-example ProsQA test set. Mean length denotes generated explicit reasoning tokens for CoT and continuous states for the latent variants.

Training paradigm	Accuracy (%) ↑	Mean length
Explicit CoT	96.60	35.96
ACT-Rec	**98.20**	8.56
ACT-Comp	**98.20**	8.69

Here, Z represents the continuous latent trajectory, and C is the retained textual rationale (which is empty in fully latent settings). Let $P(x)$ denote the formatted text preceding the latent section. To simplify the pseudocode, we define the initial prompt $Q(x)$ and the textual suffix $S(C, y)$ as follows:

$$Q(x) = P(x)\|\texttt{<start_latent>}, \qquad S(C, y) = \texttt{<end_latent>}\|C\|\texttt{\#\#\#Answer:}\|y\|\text{EOS}.$$

Notation and model operations. To formalize the model's forward pass, we use the following notations:

- E: The frozen vocabulary embeddings. All text inputs are embedded before entering the model.
- g_ψ: The trainable unconstrained latent projection layer.
- F_θ: The language model. A cached call to F_θ returns a tuple (h, ℓ, K), where h is the final-layer hidden state, ℓ represents the output logits, and K is the updated KV-cache context.
- $\ell_\mathcal{V}$: The logits restricted to the original vocabulary (whereas the full logits ℓ also include the special boundary tokens <start_latent> and <end_latent>).

E.1 Shared Inference Procedure

All paradigms share the inference procedure illustrated in Figure 1(d). The basic paradigms generate a fixed budget of latent states. In contrast, ACT variants dynamically halt latent generation either when the model predicts the <end_latent> marker or when the maximum state limit is reached.

During recurrence, the latent mapping function ρ applies either the unconstrained projection $g_\psi(h)$ or computes a softmax-weighted mixture of the top-scoring vocabulary embeddings. For unconstrained compression specifically, the predicted continuous states are rescaled by the embedding-coordinate standard deviation s_E before being fed back into the model.

Algorithm 1 Latent reasoning and answer generation

Require: Question x, trained reasoner, representation map ρ, stopping mode
$(h, \ell, K) \leftarrow F_\theta(Q(x))$
for each latent step within the inference budget **do**
  $z \leftarrow \rho(h, \ell)$
  if the model is Unconstrained Compression or ACT-Comp **then**
    $z \leftarrow s_E z$
  end if
  $(h, \ell, K) \leftarrow F_\theta(z; K)$
  if adaptive stopping is enabled and $\arg\max \ell =$ <end_latent> **then**
    break
  end if
end for
$(h, \ell, K) \leftarrow F_\theta(\texttt{<end_latent>}\|\texttt{\#\#\#Answer:}; K)$
$y \leftarrow [\,]$
for each token position within the answer budget **do**
  $v \leftarrow \arg\max \ell_\mathcal{V}$; append v to y
  if $v = \text{EOS}$ **then**
    break
  end if
  $(h, \ell, K) \leftarrow F_\theta(E_v; K)$
end for
return y

E.2 SEVEN BASIC TRAINING PARADIGMS

All reasoners are initialized from the shared explicit-CoT checkpoint, with the Reconstruction paradigms additionally initializing a separate auxiliary decoder D. During training, the forward passes retain gradients through all generated latent states to enable backpropagation. All text losses use standard next-token prediction, where the answer CE loss is computed over the final answer tokens y, excluding the preceding `<end_latent>` marker and the `###Answer:` prefix.

Algorithm 2 Unconstrained Answer

for each training minibatch (x, R, y) **do**
  $(h, \ell, K) \leftarrow F_\theta(Q(x))$
  for $t = 1, \ldots, T$ **do**
    $z_t \leftarrow g_\psi(h)$
    $(h, \ell, K) \leftarrow F_\theta(z_t; K)$
  end for
  Teacher-force $S(\varnothing, y)$ from K, obtaining suffix predictions O
  $\mathcal{L}_{\text{ans}} \leftarrow$ CE on y in O
  Backpropagate $\mathcal{L}_{\text{ans}}$ through the suffix and latent recurrence
  Update the trainable reasoner parameters
end for

Algorithm 3 Unconstrained Curriculum

for each curriculum stage s **do**
  Set latent budget T_s; reset optimizer state while retaining model weights
  for each training minibatch (x, R, y) in this stage **do**
    Split R into reasoning steps and remove the prefix covered by stage s
    $C_s \leftarrow$ remaining text, or empty text at the final stage
    $(h, \ell, K) \leftarrow F_\theta(Q(x))$
    for $t = 1, \ldots, T_s$ **do**
      $z_t \leftarrow g_\psi(h)$
      $(h, \ell, K) \leftarrow F_\theta(z_t; K)$
    end for
    Teacher-force $S(C_s, y)$ from K, obtaining suffix predictions O
    $\mathcal{L}_{\text{cot}} \leftarrow$ CE on retained rationale tokens in O
    $\mathcal{L}_{\text{ans}} \leftarrow$ CE on y in O
    Backpropagate $\mathcal{L}_{\text{cot}} + \mathcal{L}_{\text{ans}}$ and update the reasoner
  end for
end for

Algorithm 4 Constrained Curriculum

for each curriculum stage s **do**
  Set latent budget T_s; reset optimizer state while retaining model weights
  for each training minibatch (x, R, y) in this stage **do**
    Split R into reasoning steps and remove the prefix covered by stage s
    $C_s \leftarrow$ remaining text, or empty text at the final stage
    $(h, \ell, K) \leftarrow F_\theta(Q(x))$
    for $t = 1, \ldots, T_s$ **do**
      $\mathcal{V}_t \leftarrow$ highest-scoring entries of $\ell_{\mathcal{V}}$
      $z_t \leftarrow$ softmax-weighted sum of the embeddings indexed by $\mathcal{V}_t$
      $(h, \ell, K) \leftarrow F_\theta(z_t; K)$
    end for
    Teacher-force $S(C_s, y)$ from K, obtaining suffix predictions O
    Compute $\mathcal{L}_{\text{cot}}$ on C_s and $\mathcal{L}_{\text{ans}}$ on y
    Backpropagate $\mathcal{L}_{\text{cot}} + \mathcal{L}_{\text{ans}}$ and update the reasoner
  end for
end for

Algorithm 5 Unconstrained Reconstruction

for each training minibatch (x, R, y) **do**
  $(h, \ell, K) \leftarrow F_\theta(Q(x))$; $Z \leftarrow [\,]$
  for $t = 1, \ldots, T$ **do**
    $z_t \leftarrow g_\psi(h)$; append z_t to Z
    $(h, \ell, K) \leftarrow F_\theta(z_t; K)$
  end for
  Partition Z into consecutive groups $(Z_1, \ldots, Z_G)$
  for each latent group Z_g **do**
    $R_g \leftarrow$ the corresponding rationale step, or all remaining steps for the final group
    Use y as R_g if the assigned text is empty
    Teacher-force $R_g \| \mathrm{EOS}$ through D with Z_g as its prefix
    Collect decoder target-token losses as $\mathcal{E}_g$
  end for
  Compute $\mathcal{L}_{\mathrm{rec}}$ from decoder losses $\{\mathcal{E}_g\}$
  Teacher-force $S(\varnothing, y)$ from K and compute CE $\mathcal{L}_{\mathrm{ans}}$ on y
  Backpropagate $\mathcal{L}_{\mathrm{rec}} + \mathcal{L}_{\mathrm{ans}}$ through the decoder and reasoner
  Jointly update their trainable parameters
end for

Algorithm 6 Constrained Reconstruction

for each training minibatch (x, R, y) **do**
  $(h, \ell, K) \leftarrow F_\theta(Q(x))$; $Z \leftarrow [\,]$
  for $t = 1, \ldots, T$ **do**
    $\mathcal{V}_t \leftarrow$ highest-scoring entries of $\ell_{\mathcal{V}}$
    $z_t \leftarrow$ softmax-weighted sum of the embeddings indexed by $\mathcal{V}_t$; append to Z
    $(h, \ell, K) \leftarrow F_\theta(z_t; K)$
  end for
  Partition Z into consecutive groups $(Z_1, \ldots, Z_G)$
  for each latent group Z_g **do**
    $R_g \leftarrow$ the corresponding rationale step, or all remaining steps for the final group
    Use y as R_g if the assigned text is empty
    Teacher-force $R_g \| \mathrm{EOS}$ through D with Z_g as its prefix
    Collect decoder target-token losses as $\mathcal{E}_g$
  end for
  Compute $\mathcal{L}_{\mathrm{rec}}$ from decoder losses $\{\mathcal{E}_g\}$
  Teacher-force $S(\varnothing, y)$ from K and compute CE $\mathcal{L}_{\mathrm{ans}}$ on y
  Backpropagate $\mathcal{L}_{\mathrm{rec}} + \mathcal{L}_{\mathrm{ans}}$ and jointly update the reasoner and D
end for

Compression paradigms use teacher forcing for the latent trajectory. In a single causal forward pass, the outputs from the previous step (h_{j-1}, ℓ_{j-1}) are used to predict the j-th target latent.

Algorithm 7 Unconstrained Compression

for each training minibatch (x, R, y) **do**
  Tokenize R and partition it into consecutive groups $(G_1, \ldots, G_M)$
  for each token group G_j **do**
    $u_j \leftarrow \sum_{v \in G_j} E_v / \sqrt{|G_j|}$
  end for
  $U \leftarrow (u_1, \ldots, u_M)$
  Run F_θ on $Q(x) \| U \| S(\varnothing, y)$, teacher-forcing raw U and the suffix
  Extract hidden states $h_0, \ldots, h_{M-1}$ preceding the target latents
  $\widehat{u}_j \leftarrow g_\psi(h_{j-1})$ with Gaussian training perturbations
  $\mathcal{L}_{\mathrm{comp}} \leftarrow$ MSE between $\widehat{u}_j$ and u_j / s_E
  $\mathcal{L}_{\mathrm{ans}} \leftarrow$ CE on y from the same forward pass
  Backpropagate $\mathcal{L}_{\mathrm{comp}} + \mathcal{L}_{\mathrm{ans}}$ and update the reasoner
end for

Algorithm 8 Constrained Compression

for each training minibatch (x, R, y) **do**
  Tokenize R and partition it into consecutive groups $(G_1, \ldots, G_M)$
  for each token group G_j **do**
    $u_j \leftarrow \sum_{v \in G_j} E_v / \sqrt{|G_j|}$
    Retain the highest-scoring vocabulary entries of $u_j E^\top$
    $q_j \leftarrow$ softmax teacher weights on this support, with Gumbel training perturbations
    $\widetilde{z}_j \leftarrow \sum_v q_j(v) E_v$
  end for
  Run F_θ on $Q(x) \| \widetilde{Z} \| S(\varnothing, y)$ with target inputs $\widetilde{Z} = (\widetilde{z}_j)_j$
  $p_j \leftarrow \mathrm{softmax}(\ell_{j-1,\mathcal{V}})$ at each latent prediction position
  $\mathcal{L}_{\mathrm{comp}} \leftarrow D_{\mathrm{KL}}(q_j \| p_j)$ over latent positions
  $\mathcal{L}_{\mathrm{ans}} \leftarrow$ CE on y from the same forward pass
  Backpropagate $\mathcal{L}_{\mathrm{comp}} + \mathcal{L}_{\mathrm{ans}}$ and update the reasoner
end for

E.3 Adaptive Curriculum Thinking

Under the ACT paradigm, each latent state corresponds to a consecutive group of rationale tokens. Because a token group may cross the boundary between adjacent reasoning steps, it is assigned to the step containing the majority of its tokens, yielding an approximate alignment. Additionally, the token-level objective supervises tokens sampled from these groups alongside the `<end_latent>` marker.

Algorithm 9 ACT-Rec

for each curriculum stage s **do**
  Reset optimizer state, retaining the reasoner and decoder weights
  for each training minibatch (x, R, y) in this stage **do**
    Group consecutive tokens in R and assign each group to a step by token overlap
    Select the stage's groups $\mathcal{G}_s = (G_1, \ldots, G_M)$ and retained text C_s
    $(h, \ell, K) \leftarrow F_\theta(Q(x))$
    for each selected group G_j **do**
      Sample c_j uniformly from G_j; compute $e_j = \mathrm{CE}(\ell, c_j)$
      $z_j \leftarrow g_\psi(h)$
      $(h, \ell, K) \leftarrow F_\theta(z_j; K)$
      Teacher-force $G_j \| \mathrm{EOS}$ through D with the single state z_j as its prefix
      Collect decoder target-token losses as $\mathcal{E}_j$
    end for
    $e_{\mathrm{end}} \leftarrow \mathrm{CE}(\ell, \texttt{<end_latent>})$
    Compute $\mathcal{L}_{\mathrm{tok}}$ from sampled-token losses $\{e_j\}$ and e_{end}
    Compute $\mathcal{L}_{\mathrm{rec}}$ from decoder losses $\{\mathcal{E}_j\}$
    Teacher-force $S(C_s, y)$ from K, obtaining suffix predictions O
    Compute $\mathcal{L}_{\mathrm{cot}}$ on C_s and $\mathcal{L}_{\mathrm{ans}}$ on y
    Backpropagate $\mathcal{L}_{\mathrm{rec}} + \mathcal{L}_{\mathrm{tok}} + \mathcal{L}_{\mathrm{cot}} + \mathcal{L}_{\mathrm{ans}}$
    Jointly update the trainable reasoner and decoder parameters
  end for
end for

Algorithm 10 ACT-Comp

for each curriculum stage s **do**
  Reset optimizer state, retaining the reasoner weights
  for each training minibatch (x, R, y) in this stage **do**
    Group consecutive tokens in R and assign each group to a step by token overlap
    Select the stage's groups $\mathcal{G}_s = (G_1, \ldots, G_M)$ and retained text C_s
    for each selected group G_j **do**
      $u_j \leftarrow \sum_{v \in G_j} E_v / \sqrt{|G_j|}$; sample c_j uniformly from G_j
    end for
    $U \leftarrow (u_1, \ldots, u_M)$
    Run F_θ on $Q(x) \| U \| S(C_s, y)$, teacher-forcing raw U and the suffix
    Read $h_0, \ldots, h_M$ and $\ell_0, \ldots, \ell_M$ along the target latent prefix
    $\widehat{u}_j \leftarrow g_\psi(h_{j-1})$ with Gaussian training perturbations
    $\mathcal{L}_{\text{comp}} \leftarrow$ MSE between $\widehat{u}_j$ and u_j / s_E
    $e_j \leftarrow \text{CE}(\ell_{j-1}, c_j)$; $e_{\text{end}} \leftarrow \text{CE}(\ell_M, \texttt{<end_latent>})$
    Compute $\mathcal{L}_{\text{tok}}$ from sampled-token losses $\{e_j\}$ and e_{end}
    Compute $\mathcal{L}_{\text{cot}}$ on C_s and $\mathcal{L}_{\text{ans}}$ on y
    Backpropagate $\mathcal{L}_{\text{comp}} + \mathcal{L}_{\text{tok}} + \mathcal{L}_{\text{cot}} + \mathcal{L}_{\text{ans}}$
    Update the trainable reasoner parameters
  end for
end for

E.4 Probe Training

Each trained reasoner is paired with an independent probe q_ϕ that reconstructs the reference rationale R from its full latent trajectory Z. The probe context $\Pi(Z)$ inserts these continuous states into the following template:

```
Convert the following latent reasoning into explicit text:
###Z###
Converted text:
```

Algorithm 11 Probe training for rationale reconstruction

Require: Trained reasoner, training set $\mathcal{D}_{\text{train}}$
Freeze the reasoner and switch it to inference mode
Initialize the trajectory cache $\mathcal{C}$
for each minibatch (x, R, y) in $\mathcal{D}_{\text{train}}$ **do**
  Generate Z from $Q(x)$ using the latent recurrence in Algorithm 1
  Cache (Z, R) in $\mathcal{C}$
end for
Initialize the probe q_ϕ from the base language model
for each training epoch **do**
  for each minibatch (Z, R) in $\mathcal{C}$ **do**
    Insert Z as continuous inputs in the probe context $\Pi(Z)$
    Teacher-force $R \| \text{EOS}$ through q_ϕ with context $\Pi(Z)$
    $\mathcal{L}_{\text{probe}} \leftarrow$ CE on R
    Backpropagate $\mathcal{L}_{\text{probe}}$ and update ϕ
  end for
end for
return the trained probe q_ϕ