Latent-SFT: LLM Latent Reasoning as Chain of Superposition

Anonymous ACL submission

Abstract

Latent reasoning offers a computation-efficient alternative to Chain-of-Thought but often suffers from performance degradation due to distributional misalignment and ambiguous chain definitions. Ideally, latent reasoning should function as a superposition of multiple reasoning paths. To realize this, we introduce Latent-SFT, a unified framework addressing challenges at three levels: token, chain, and learning. First, we define the Latent-Vocab to constrain hidden states within the pre-trained vocab-space. Second, we construct the Latent-Chain via Induction-Supervision Masking to ensure semantic compactness and sufficiency. Third, we employ Latent-Optim with stochastic Gumbel-Softmax to guide the model toward generalizable solutions. Empirical results demonstrate that Latent-SFT consistently outperforms explicit SFT across six mathematical benchmarks (e.g., GSM8k, AIME24) while achieving a 2.7x to 5.5x reduction in reasoning length. Analysis confirms that our method effectively captures a superposition of diverse reasoning trajectories rather than merely compressing a single path.

1 Introduction

Large language models (LLMs) have shown strong reasoning abilities (Zhao and Zhang, 2024) across diverse tasks (Cao and Zhao, 2025; Wei et al., 2025). A key driver of this success is chain-of-thought prompting (Wei et al., 2022; Xu et al., 2024; Yao et al., 2023), which solves complex problems by generating intermediate steps in natural language. However, current reasoning models (Team, 2025; DeepSeek-AI, 2025) rely on long chains of tokens, leading to redundancy and high computational cost. Latent Reasoning is a promising paradigm that significantly reduces the length of reasoning chains by shifting computation to a continuous latent space. This paradigm theoretically enhances information bandwidth and exploration efficiency (Zhu et al., 2025), yet current methods frequently suffer from severe performance degradation compared to explicit reasoning (Hao et al., 2024).

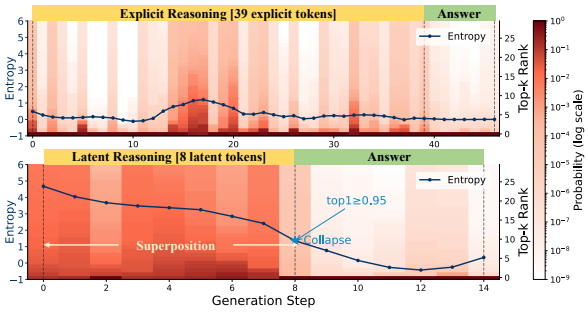


Figure 1: Comparison of Latent and Explicit Reasoning. Latent reasoning operates as a superposition state, where each step carries a higher entropy. Therefore, it can complete the reasoning process and derive the correct answer using fewer tokens.

Ideally, latent reasoning should function as a superposition of multiple reasoning paths, enabling the model to explore diverse logical trajectories simultaneously within a latent space. However, realizing this potential is impeded by three hierarchical flaws in current methodologies. First, at the **latent token level**, existing approaches rely on raw, unconstrained hidden states (Hao et al., 2024; Cheng and Durme, 2024). These high-dimensional vectors suffer from severe distributional shifts relative to the model's vocab-space, making them difficult for the model to interpret. Second, at the **latent chain level**, current definitions are ambiguous; they typically assume the latent chain need only decode the final answer (Shen et al., 2025). This "black box" assumption neglects intermediate semantic fidelity, failing to capture the step-by-step logic required for complex deduction. Third, at the **latent learning level**, optimization is inefficient. Models are updated solely via indirect backpropagation from the final explicit supervision (Hao et al., 2024), lacking

the step-by-step guidance necessary to learn stable reasoning trajectories.

To systematically address these challenges, we introduce **Latent-SFT**, a comprehensive framework that unifies token-level representation, chain-level definition, and learning strategies. For token-level representation, we formally define the **Latent-Vocab** as a superposition within the vocab-space (Section 4). This formulation aligns latent tokens with the model's inherent vocabulary distribution. For the chain-level definition, we introduce the **Latent-Chain** (Section 5), constructed via Induction-Supervision Masking to satisfy semantic compactness, compatibility, and sufficiency. Finally, regarding the learning strategy, to enable the LLM to autonomously generate these latent chains, we propose **Latent-Optim**. Within this framework, we address the static nature of the constructed latent chains by employing stochastic Gumbel-Softmax optimization. This technique injects Gumbel noise to approximate sampling, acting as an implicit Hessian regularizer (see Theorem 1 in Appendix F) that guides the model toward flatter, more generalizable minima.

As illustrated in Figure 1, our model realizes the ideal latent reasoning via a chain of superposition analogous to a quantum wavefunction. Upon reasoning convergence, the system undergoes entropy collapse, precipitating the latent superposition into a discrete linguistic eigenstate. Empirical results demonstrate that Latent-SFT consistently outperforms explicit SFT models across six diverse mathematical reasoning benchmarks. Notably, it achieves a 2.7x to 5.5x reduction in reasoning chain length while surpassing prior latent reasoning approaches. Further analysis reveals that its latent reasoning is not merely a compression of a single chain, but rather a superposition of multiple reasoning chains.

2 Related Work

Current research remains entangled in debates over the reasonable representation of latent tokens, while the critical dimensions of latent chain definition and optimization strategies have been largely overlooked. Consequently, the definition of the latent token serves as the primary differentiator in this field. We categorize existing research based on this pivotal factor.

Unconstrained Representations. Pioneering approaches, such as COCONUT (Hao et al., 2024) and CCOT (Cheng and Durme, 2024), adopted a direct approach by utilizing the last-layer hidden states as latent tokens. While computationally straightforward, these methods rely on raw, high-dimensional vectors that are unconstrained by the model's vocabulary distribution. Although subsequent works like CODI (Shen et al., 2025) and PCCOT (Wu et al., 2025b) attempted to mitigate this misalignment via MLP mapping or distillation, they fundamentally remain bound to the continuous hidden state paradigm. Consequently, without strict constraints aligned with the pre-trained semantic manifold, these methods often face optimization instability and limited performance gains compared to explicit reasoning.

Constrained Representations. To address the distributional gap, recent studies have introduced structural constraints. CoLaR (Tan et al., 2025) represents a step forward by modeling latent tokens via Gaussian distributions, achieving improved alignment through a dedicated latent head. Similarly, methods like Soft Thinking (Zhang et al., 2025), Mixture-of-Input (MoI) (Zhuang et al., 2025), and recent vocabulary-space superposition approaches (Zeng et al., 2025; Butt et al., 2025) attempt to bridge the gap by constructing latent tokens as linear combinations of vocabulary embeddings. However, their performance gains on reasoning tasks remain marginal. In contrast, we formally define the Latent-Vocab and Latent-Chain to enable simultaneous multi-path exploration, thereby significantly shortening reasoning chains while surpassing existing baselines.

3 Latent-SFT

Latent-SFT consists of three coupled components. First, **Latent-Vocab** defines each latent token as a simplex-constrained vocabulary distribution $\boldsymbol{\alpha}$ and its corresponding superposition embedding $\mathbf{z} = \mathbf{E}\boldsymbol{\alpha}$. Second, **Latent-Chain** (Phase A in Figure 2) converts explicit CoT trajectories into static latent-chain targets $\{(\boldsymbol{\alpha}_i, \mathbf{z}_i)\}_{i=1}^{N}$ through induction-supervision masking. Third, **Latent-Optim** (Phase B) trains the LLM to generate these latent-chain distributions directly. At inference time, the construction encoder is not used; the trained LLM autoregressively produces latent-token distributions and then decodes the final answer. The complete algorithmic flow is summarized in Algorithm 1.

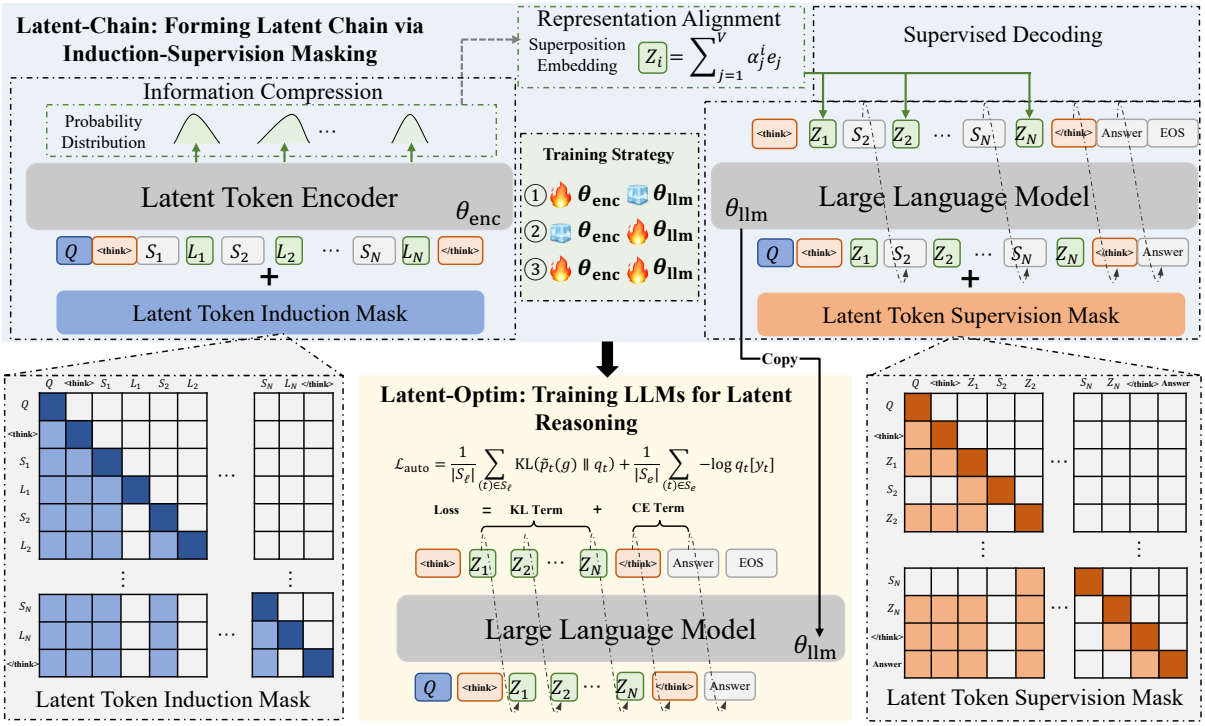


Figure 2: Overview of Latent-SFT. **Latent-Vocab** defines each latent token as a vocabulary distribution $\boldsymbol{\alpha}_i$ and its superposition embedding $\mathbf{z}_i$. **Latent-Chain** in Phase (a) constructs static latent-chain targets $\{(\boldsymbol{\alpha}_i, \mathbf{z}_i)\}$ from explicit CoT trajectories via Induction–Supervision Masking. **Latent-Optim** in Phase (b) trains the LLM to generate such latent chains directly using KL and CE supervision. At inference time, only the trained LLM is used to produce latent-token distributions and decode the final answer.

4 Latent-Vocab: Superposition of Vocab-Space

Prior studies have explored both unconstrained hidden states and several constrained alternatives for latent tokens, yet it remains unclear what kind of constraint best aligns latent reasoning with the pre-trained language-model interface. We therefore conduct two preliminary analyses addressing the fundamental question: *what is the reasonable representation for latent tokens?*

4.1 The Distributional Discrepancy in Vector Spaces

Most existing studies (Shen et al., 2025; Wu et al., 2025b) follow the COCONUT design and use last-layer hidden states directly as latent tokens. However, statistical analysis in representative LLMs reveals a critical discrepancy between these vector spaces.

Observation 1. *Significant distributional divergence exists between the last-layer hidden states and the token embedding space.* As illustrated in Figure 3 and Figure 7, PCA projections demonstrate a substantial separation between the manifolds of hidden states and token embeddings. Quantitative analysis confirms this divergence, showing significant disparities in first and second-order statistics (mean and variance), as well as large inter-distribution distances. From a model optimization perspective, shallow parameters (e.g., RMSNorm) are calibrated exclusively on the token embedding distribution during pre-training. Consequently, using raw hidden states as latent tokens can introduce distribution shift and training instability. We further find that this gap is not reduced, but enlarged, in the released COCONUT checkpoint (Appendix C),

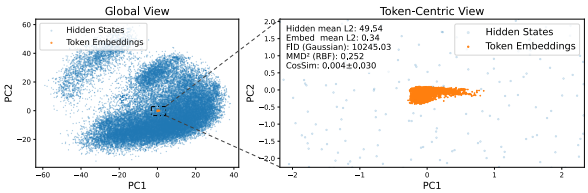


Figure 3: Visualization of last-layer hidden states and token embeddings of the LLaMA-3.2-1B-Instruct model on the GSM8k dataset. Results are shown from both global and token-specific perspectives to highlight the substantial distributional differences. In addition, we report the statistical differences between the two distributions using the Fréchet Inception Distance (FID) and the squared Maximum Mean Discrepancy (MMD^2). We also estimate cosine similarity via random sampling.

suggesting that hidden-state latent tokens may remain misaligned with the pretrained vocabulary interface after latent-reasoning training.

4.2 The Structured Anisotropy of Vocab-Space

Building on Observation 1, we posit that latent tokens should preserve the pretrained geometry of vocab-space. Since LLMs are trained through discrete next-token prediction, their semantic interface is not an unconstrained continuous Gaussian space, but a vocabulary distribution mapped through token embeddings and the language modeling head. To examine this geometry, we analyze the singular value spectra of token embedding matrices across various LLMs.

Observation 2. *The vocabulary embedding space is structured and anisotropic rather than an unconstrained isotropic vector space.* The singular value decay curves in Figure 6 (Appendix D) show that spectral energy is distributed unevenly across directions. Rather than implying a low-dimensional collapse, this suggests that latent tokens should inherit the structured vocabulary interface induced by discrete language modeling, instead of being modeled as free hidden states or Gaussian latent variables.

These findings imply that latent tokens should be represented through the same vocabulary interface used during pretraining. Motivated by this insight, we propose the following definition.

Definition 1 (*Superposition Embedding*)**.** We define a latent token $z \in \mathbb{R}^d$ as a simplex-constrained superposition over vocab-space. Formally, let $\mathbf{E} = [e_1, e_2, \ldots, e_V] \in \mathbb{R}^{d \times V}$ denote the token embedding matrix. A latent token z is expressed as a probability-weighted mixture of token embeddings:

$$z = \mathbf{E}\boldsymbol{\alpha} = \sum_{i=1}^{V} \alpha_i e_i, \tag{1}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_V]^\top \in \Delta^{V-1}$ is a token-probability vector. We do not require a unique linear decomposition; the key is a vocabulary-aligned parameterization compatible with the LM head and the pretrained discrete-token interface.

5 Latent-Chain: Forming Latent Chain via Induction-Supervision Masking

Having established Latent-Vocab, we address the next core question: *what are the requisite properties of the higher-level Latent-Chain?* To avoid treating latent reasoning as a black box optimized only for the final answer, we require three semantic criteria: (1) Semantic Compactness, compressing multiple explicit tokens into one latent representation; (2) Semantic Compatibility, adhering to the vocab-space defined in Section 4; and (3) Semantic Sufficiency, retaining enough information for subsequent reasoning. Figure 2 implements these criteria through information compression, representation alignment, and supervised decoding.

Information Compression Process (Semantic Compactness). We insert special tokens into the explicit reasoning sequence to segment and capture intermediate reasoning information. Given an input $X = [Q, \texttt{<think>}, C, \texttt{</think>}]$, we define a segmentation function $\mathrm{Seg}(\cdot)$ that partitions the explicit reasoning chain C into N subsegments, denoted as $C = \{S_i\}_{i=1}^{N}$. This segmentation can be governed either by a fixed compression ratio r or by semantic boundaries; in our experiments, we use the fixed-length version detailed in Appendix G.1. Following each subsegment S_i, we insert a special latent placeholder L_i, which is a newly added special token in the vocabulary. Crucially, we apply a *Latent Token Induction Mask* (LTIM) to restrict the attention of L_i exclusively to the preceding explicit reasoning context (see Figure 2). This design ensures that the semantic content encoded by each latent token represents a cumulative summary of the reasoning process up to that point.

Representation Alignment Process (Semantic Compatibility). For each inserted placeholder L_i, we extract its hidden state $\mathbf{h}_i \in \mathbb{R}^d$ from the model under LTIM masking. To strictly enforce **Semantic Compatibility**, we map $\mathbf{h}_i$ to a vocabulary distribution through the language modeling head and reconstruct it as a mixture of token embeddings. Let $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_V] \in \mathbb{R}^{d \times V}$ denote the token embedding matrix and $\mathbf{W} \in \mathbb{R}^{d \times V}$ be the language modeling head (where $\mathbf{W} = \mathbf{E}$ under weight tying). We compute the vocabulary logits and the resulting probability distribution as:

$$\boldsymbol{\ell}_i = \mathbf{W}^\top \mathbf{h}_i, \quad \boldsymbol{\alpha}_i = \mathrm{softmax}\left(\frac{\boldsymbol{\ell}_i}{\tau}\right) \in \Delta^{V-1}, \tag{2}$$

where $\tau > 0$ is the temperature parameter. The final latent representation is then obtained via linear interpolation:

$$\mathbf{z}_i = \mathbf{E}\boldsymbol{\alpha}_i = \sum_{v=1}^{V} \alpha_{i,v} \mathbf{e}_v. \tag{3}$$

This operation instantiates Definition 1, guaranteeing that $\mathbf{z}_i$ resides within the convex hull of the embedding space $\{\mathbf{e}_v\}_{v=1}^{V}$. Consequently, latent tokens remain confined to the intrinsic geometry of the LLM's vocabulary manifold. During the subsequent optimization phase, the distribution $\boldsymbol{\alpha}_i$ serves as the supervision target (via KL divergence) for the latent positions.

Supervised Decoding Process (Semantic Sufficiency). Given the latent representations $\{\mathbf{z}_i\}_{i=1}^{N}$ from the alignment step, we use each $\mathbf{z}_i$ to decode the content from the $(i+1)$-th explicit reasoning subsegment through to the final answer. Let the prefix up to step i be $\Pi_i = [Q, \texttt{<think>}, \mathbf{z}_1, S_2, \mathbf{z}_2, \ldots, \mathbf{z}_i]$. Here S_1 is not used as a supervised decoding label because it has already been encoded into $\mathbf{z}_1$ by LTIM. Under the Latent Token Supervision Mask (LTSuM), tokens in $\mathcal{Y}_i = [S_{i+1}, \ldots, S_N, \texttt{</think>}, \mathrm{Answer}]$ are decoded conditioned only on Π_i (in particular on $[Q, \texttt{<think>}, \mathbf{z}_1, ..., \mathbf{z}_i]$), while attention to future latent tokens and to past explicit reasoning steps is blocked. Denoting by $\mathcal{J}_i$ the index set of tokens in $\mathcal{Y}_i$, the supervised decoding objective is

$$\mathcal{L}_{\text{sup}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\mathcal{J}_i|}\sum_{t\in\mathcal{J}_i} \Big(-\log p_\theta\big(x_t \,\big|\, \Pi_i;\ \text{LTSuM}\big)\Big). \tag{4}$$

Having defined the three semantic properties above, we next describe how to optimize the latent-token encoder and the target LLM to construct high-quality latent chains in practice.

Latent-Chain Construction Training. The strict LTIM/LTSuM constraints can destabilize naive end-to-end SFT. We therefore adopt a progressive schedule: we first optimize the latent-token encoder, then the LLM backbone, and finally fine-tune both components jointly under $\mathcal{L}_{\text{sup}}$. This EM-style staging lets each module adapt to latent-token induction and supervised decoding before full co-adaptation, stabilizing training under the distributional shift introduced by latent tokens. Full details are provided in Appendix G.

6 Latent-Optim: Training LLMs for Latent Reasoning

Having constructed high-quality Latent-Chains in Section 5, we next train the LLM to generate them autonomously during reasoning. Latent-Optim provides explicit supervision for intermediate latent positions, avoiding reliance on indirect backpropagation from the final answer alone while improving generalization.

We concatenate latent chains with the explicit context to form the training sequence $X = [Q, \texttt{<think>}, \mathbf{z}_1, \ldots, \mathbf{z}_N, \texttt{</think>}, \mathrm{Answer}]$. The explicit segments S_i are removed in this stage because they are used only to supervise the latent-token encoder during Phase A; here each $\mathbf{z}_i = \mathbf{E}\boldsymbol{\alpha}_i$ is a base latent target constructed by the trained encoder. A straightforward optimization approach is to minimize the KL divergence between the static latent chain $\boldsymbol{\alpha}_t$ and the model's predicted vocabulary distribution $\mathbf{q}_t = \mathrm{softmax}(\mathbf{W}^\top \mathbf{h}_t)$ at latent position t, combined with the standard cross-entropy loss for explicit answer tokens S_{exp}:

$$\mathcal{L}_{\text{static}} = \frac{1}{|S_{\text{lat}}|}\sum_{t\in S_{\text{lat}}} \mathrm{KL}(\boldsymbol{\alpha}_t \,||\, \mathbf{q}_t) + \frac{1}{|S_{\text{exp}}|}\sum_{t\in S_{\text{exp}}} \big(-\log q_t[y_t]\big). \tag{5}$$

However, directly supervising the LLM with static latent chains $\boldsymbol{\alpha}_t$ can lead to overfitting on the specific artifacts of the construction encoder, thereby limiting the model's generalization capability.

To address this, we propose a **Stochastic Latent Optimization** strategy. Instead of treating the latent targets as immutable labels, we view them as parameters of a generative distribution. During each training epoch, we dynamically sample a perturbed latent distribution using the Gumbel-Softmax trick (Wu et al., 2025a). Specifically, for a latent position $t \in S_{\text{lat}}$, let $\boldsymbol{\alpha}_t$ be the static latent chain. We construct a stochastic distribution $\tilde{\mathbf{p}}_t$ and its corresponding latent input $\tilde{\mathbf{z}}_t$ by injecting Gumbel noise:

$$\begin{aligned} \tilde{\mathbf{p}}_t &= \mathrm{softmax}\left(\frac{\log \boldsymbol{\alpha}_t + \mathbf{g}_t}{\tau}\right), \\ \tilde{\mathbf{z}}_t &= \mathbf{E}\tilde{\mathbf{p}}_t, \\ \mathbf{g}_t &\sim \mathrm{Gumbel}(0, 1), \end{aligned} \tag{6}$$

In our experiments, we set the temperature $\tau = 1$. The noise vector $\mathbf{g}_t$ is resampled independently at every training step and applied to both the KL supervision target $\tilde{\mathbf{p}}_t$ and the teacher-forced latent input $\tilde{\mathbf{z}}_t$.

Consequently, the final optimization objective is reformulated as the expectation over this noise:

$$\mathcal{L}_{\text{auto}}(\theta_{\text{llm}}) = \frac{1}{|S_{\text{lat}}|} \sum_{t \in S_{\text{lat}}} \mathbb{E}_{\mathbf{g}} \left[\text{KL}\big(\tilde{\mathbf{p}}_t(\mathbf{g}) \,||\, \mathbf{q}_t\big) \right] + \frac{1}{|S_{\text{exp}}|} \sum_{t \in S_{\text{exp}}} \big(-\log q_t[y_t] \big). \tag{7}$$

By minimizing the expectation over the Gumbel noise, we effectively smooth the optimization landscape. As we formally prove in **Theorem 1** (see Appendix F), this stochastic injection acts as an implicit regularizer that penalizes the sharpness of the loss Hessian, thereby guiding the model toward flatter minima and enhancing generalization across diverse reasoning paths.

Inference. At inference time, the Latent-Chain construction encoder is discarded. Given a question Q, the trained LLM first enters the latent reasoning stage after `<think>`. At each step t, the model produces a hidden state $\mathbf{h}_t$ and maps it to a vocabulary distribution $\boldsymbol{\alpha}_t = \text{softmax}(\mathbf{W}^\top \mathbf{h}_t)$. We first decode a candidate discrete token from $\boldsymbol{\alpha}_t$; if it is `</think>`, we append the discrete stop token and terminate latent mode without appending a latent token at this position. Otherwise, we feed the corresponding superposition embedding $\mathbf{z}_t = \mathbf{E}\boldsymbol{\alpha}_t$ as the next latent input. Thus, the hidden state is used only to parameterize a vocabulary distribution through the LM head; we do not feed unconstrained hidden states as latent tokens. The number of latent steps T is determined dynamically by the first generated `</think>` token, or by the maximum generation length if no stop token is generated. We report # L as the number of generated latent superposition tokens.

7 Experiments

7.1 Experimental Setups

Datasets. To comprehensively assess the effectiveness of our method, we conducted experiments on both low-difficulty mathematical reasoning datasets (GSM8k-Aug (Deng et al., 2024), GSM-Hard (Gao et al., 2023), SVAMP (Patel et al., 2021), and MultiArith (Roy and Roth, 2015)) and high-difficulty datasets (Math500 (Hendrycks et al., 2021) and AIME24). For the low-difficulty tasks, we trained on the GSM8k-Aug training split to ensure fair comparison with prior work. For the high-difficulty tasks, due to limited computational resources, we used a subset of Open-R1 (Hugging Face, 2025) containing explicit reasoning chains shorter than 4k tokens, resulting in 57,544 training samples. This restriction inevitably constrained the performance of our method.

Baselines and Models. The baselines considered in our experiments include COT-SFT (Wei et al., 2022), iCOT (Deng et al., 2024), COCONUT (Hao et al., 2024), CODI (reproduced) (Tan et al., 2025), and CoLaR (Tan et al., 2025). To align model capabilities with task complexity, we strictly match base models to dataset difficulty. For low-difficulty tasks, we adopt LLaMA-3.2-1B-Instruct and LLaMA-3.2-3B-Instruct (Dubey and et al., 2024), while for high-difficulty tasks, we employ DeepSeek-Distill-Qwen-7B (DeepSeek-AI, 2025; Yang et al., 2025). We report Pass@1, reasoning length (# L), and their ratio Pass@1/# L; full evaluation protocol and implementation details are provided in Appendix H.

7.2 Main Results

Performance on Low-Difficulty Tasks. Table 1 compares Latent-SFT with baselines under different compression ratios r. Latent-SFT consistently improves Pass@1 and shortens reasoning chains relative to latent-reasoning baselines, and generalizes well to all three out-of-distribution datasets. On the 1B model, Latent-SFT(2) improves average Pass@1 by 2.0% while reducing chain length by approximately 5.5×, outperforming explicit CoT-SFT on nearly all datasets. This suggests that explicit reasoning does not represent the upper bound of latent reasoning: with an appropriate definition and learning strategy for latent tokens, latent reasoning can exceed explicit reasoning in accuracy. A real reasoning example is provided in Figure 10 (Appendix).

Performance on High-Difficulty Reasoning Tasks. Table 2 shows that the same trend extends to competition-level math on DeepSeek-Distill-Qwen-7B: Latent-SFT with *Superposition Embedding* surpasses CoT-SFT on Pass@1 while shortening chains by 2.7–3.6×, including on AIME24 where prior latent methods often collapse. It also clearly outperforms the hidden-state variant, indicating that vocab-space alignment remains important at larger scale.

Table 1: Experimental results on four low-difficulty datasets. The number in brackets represents the compression ratio r. " " marks the best result, and " " marks the second best. - Hidden State refers to removing the Representation Alignment step and directly using the last-layer hidden state as the representation of latent tokens. - w/o Gumbel indicates removing the stochastic Gumbel noise injection, effectively performing deterministic optimization on static soft labels. - w/o LTIM and - w/o LTSuM denote removing the Latent Token Induction Mask and Latent Token Supervision Mask, respectively, reducing the model to a standard autoregressive attention mechanism.

	GSM8k-Aug		GSM-Hard		SVAMP		MultiArith		Average		
	Pass@1 ↑	# L ↓	Pass@1 ↑	# L ↓	Pass@1 ↑	# L ↓	Pass@1 ↑	# L ↓	Pass@1 ↑	# L ↓	Pass@1/# L ↑
LLaMA-3.2-1B-Instruct											
CoT-SFT	49.4$_{\pm.6}$	70.1$_{\pm.2}$	12.1$_{\pm.2}$	78.4$_{\pm.1}$	60.5$_{\pm.4}$	37.2$_{\pm.1}$	93.2$_{\pm.5}$	38.5$_{\pm.1}$	53.8	56.5	0.95
iCoT	19.8$_{\pm.2}$	0.00$_{\pm.0}$	3.87$_{\pm.2}$	0.00$_{\pm.0}$	36.4$_{\pm.5}$	0.00$_{\pm.0}$	38.2$_{\pm.7}$	0.00$_{\pm.0}$	24.6	0.00	-
COCONUT	23.1$_{\pm.3}$	6.00$_{\pm.0}$	5.49$_{\pm.3}$	6.00$_{\pm.0}$	40.7$_{\pm.7}$	6.00$_{\pm.0}$	41.1$_{\pm.2}$	6.00$_{\pm.0}$	27.6	6.00	4.60
CODI	13.3$_{\pm.6}$	6.00$_{\pm.0}$	2.97$_{\pm.2}$	6.00$_{\pm.0}$	21.7$_{\pm.7}$	6.00$_{\pm.0}$	19.2$_{\pm.8}$	6.00$_{\pm.0}$	14.3	6.00	2.38
CoLaR-5	26.8$_{\pm.2}$	5.57$_{\pm.0}$	5.87$_{\pm.1}$	6.53$_{\pm.0}$	48.4$_{\pm.5}$	2.95$_{\pm.0}$	86.4$_{\pm.4}$	3.21$_{\pm.0}$	41.7	4.57	9.12
CoLaR-2	40.1$_{\pm.2}$	12.7$_{\pm.0}$	9.08$_{\pm.0}$	14.0$_{\pm.0}$	54.9$_{\pm.2}$	6.11$_{\pm.0}$	91.3$_{\pm.1}$	7.35$_{\pm.0}$	48.8	10.0	4.88
Latent-SFT(4)	46.8$_{\pm.1}$	6.81$_{\pm.0}$	11.2$_{\pm.2}$	7.77$_{\pm.2}$	57.6$_{\pm.3}$	3.60$_{\pm.1}$	94.0$_{\pm.2}$	4.15$_{\pm.0}$	52.4	5.58	**9.39**
- w/o Gumbel	44.6$_{\pm.0}$	6.64$_{\pm.0}$	10.2$_{\pm.0}$	7.68$_{\pm.0}$	55.6$_{\pm.1}$	3.33$_{\pm.0}$	93.6$_{\pm.0}$	4.11$_{\pm.0}$	51.0	5.44	9.37
- Hidden State	36.7$_{\pm.4}$	6.61$_{\pm.1}$	8.41$_{\pm.2}$	7.77$_{\pm.0}$	44.4$_{\pm.2}$	3.22$_{\pm.0}$	90.3$_{\pm.1}$	4.06$_{\pm.0}$	45.0	5.41	8.32
- w/o LTIM	38.8$_{\pm.1}$	6.59$_{\pm.1}$	8.67$_{\pm.1}$	8.01$_{\pm.1}$	48.9$_{\pm.1}$	3.34$_{\pm.1}$	90.8$_{\pm.0}$	4.36$_{\pm.0}$	46.8	5.58	8.39
- w/o LTSuM	42.6$_{\pm.0}$	6.70$_{\pm.0}$	9.98$_{\pm.1}$	7.60$_{\pm.0}$	53.9$_{\pm.1}$	3.36$_{\pm.0}$	92.6$_{\pm.0}$	4.27$_{\pm.0}$	49.7	5.48	9.07
Latent-SFT(2)	**52.4**$_{\pm.2}$	12.8$_{\pm.0}$	**12.6**$_{\pm.4}$	15.0$_{\pm.1}$	**61.3**$_{\pm.4}$	6.21$_{\pm.1}$	**96.8**$_{\pm.3}$	7.19$_{\pm.2}$	**55.8** $_{+2.0}$	10.3 $_{5.5\times}$	5.41
- w/o Gumbel	50.4$_{\pm.1}$	12.4$_{\pm.0}$	10.9$_{\pm.0}$	13.9$_{\pm.0}$	57.8$_{\pm.1}$	5.98$_{\pm.0}$	93.8$_{\pm.0}$	7.21$_{\pm.0}$	53.2	9.87	5.39
- Hidden State	41.1$_{\pm.3}$	12.8$_{\pm.0}$	9.01$_{\pm.1}$	14.2$_{\pm.1}$	49.6$_{\pm.2}$	6.17$_{\pm.0}$	91.1$_{\pm.0}$	7.29$_{\pm.0}$	47.7	10.1	4.72
- w/o LTIM	41.3$_{\pm.1}$	12.9$_{\pm.1}$	9.32$_{\pm.0}$	14.0$_{\pm.0}$	53.6$_{\pm.1}$	6.25$_{\pm.1}$	91.5$_{\pm.0}$	7.96$_{\pm.0}$	48.9	10.3	4.75
- w/o LTSuM	49.0$_{\pm.0}$	12.1$_{\pm.0}$	10.3$_{\pm.0}$	13.7$_{\pm.0}$	56.5$_{\pm.2}$	6.22$_{\pm.0}$	93.0$_{\pm.0}$	7.23$_{\pm.0}$	52.2	9.81	5.32
LLaMA-3.2-3B-Instruct											
CoT-SFT	61.1$_{\pm.3}$	70.5$_{\pm.2}$	**18.8**$_{\pm.2}$	80.6$_{\pm.1}$	70.9$_{\pm.2}$	34.2$_{\pm.1}$	**98.3**$_{\pm.5}$	37.4$_{\pm.1}$	62.2	55.7	1.12
Latent-SFT(2)	**62.4**$_{\pm.1}$	12.5$_{\pm.3}$	18.3$_{\pm.3}$	14.6$_{\pm.2}$	**71.3**$_{\pm.7}$	5.94$_{\pm.1}$	98.7$_{\pm.4}$	7.15$_{\pm.1}$	**62.7** $_{+0.5}$	10.3 $_{5.6\times}$	**5.39**

Table 2: Experimental results of DeepSeek-Distill-Qwen-7B on high-difficulty datasets. We primarily examine the performance gap between *Superposition Embedding* and hidden state representations. To ensure a fair comparison, we set the maximum reasoning chain length to 4K tokens for all models.

Model	MATH-500			AIME24		
	Pass@1 ↑	# L ↓	Pass@1/# L × 100 ↑	Pass@1 ↑	# L ↓	Pass@1/# L × 100 ↑
COT-SFT	73.8$_{\pm1.32}$	2766$_{\pm319}$	2.67	17.7$_{\pm1.23}$	4016$_{\pm343}$	0.44
Latent-SFT (Hidden State)	67.8$_{\pm0.94}$	698.1$_{\pm22.1}$	9.71	7.78$_{\pm2.08}$	1509$_{\pm114}$	0.51
Latent-SFT (*Superposition Embedding*)	**79.8**$_{\pm1.35}$ $_{+6.0}$	759.3$_{\pm36.9}$ $_{3.6\times}$	**10.51**	**19.2**$_{\pm1.60}$ $_{+1.5}$	1484$_{\pm420}$ $_{2.7\times}$	**1.29**

7.3 Ablation Study

Necessity of Masking Mechanisms. LTIM enforces causal independence to stabilize training, while LTSuM ensures latent tokens retain sufficient semantic information for decoding. As shown in Table 1, removing either mechanism leads to a significant accuracy drop (∼5%), confirming their critical role in maintaining the structural integrity of the reasoning chain.

Latent Representation and Stochastic Optimization. We further validate our specific design choices. First, replacing *Superposition Embedding* with raw hidden states degrades performance, demonstrating the importance of aligning latent tokens with the intrinsic vocabulary manifold. Second, ablating Gumbel-noise injection (w/o Gumbel) leads to a noticeable decline, confirming that stochasticity acts as a vital regularizer (Theorem 1) to prevent overfitting and enhance generalization.

8 Essence of Latent Reasoning

A central premise of Latent-SFT is that latent reasoning should go beyond single-chain compression and support *reasoning superposition*. We therefore examine whether the learned latent process preserves information from a single explicit chain while also capturing multiple reasoning paths, using two complementary hypotheses.

Hypothesis 1. Latent Reasoning as the Compression of a Single Reasoning Chain. To quantify how much explicit information is preserved at each step of latent reasoning, we introduce the **Top-K Effective Compression Ratio (ECR@K)**. This metric measures the proportion of explicit chain tokens that fall within the Top-K of the probability distribution at the corresponding latent step. Larger values indicate stronger horizontal compression across the reasoning chain, while values greater than one reflect true compression of the explicit

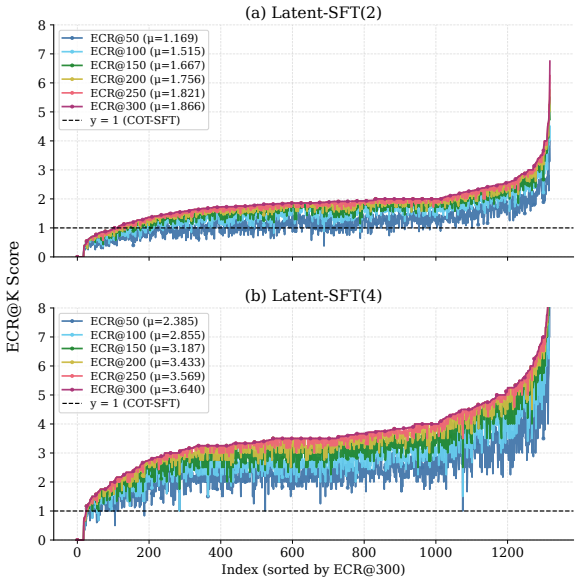


Figure 4: ECR@K analysis of Latent-SFT on the GSM8k-Aug test set. Each curve represents the per-sample ECR@K, with samples sorted along the x-axis by their ECR@300 values. The dashed line indicates the baseline for uncompressed single-chain reasoning. Latent-SFT consistently surpasses this baseline, with higher training compression ratios yielding greater information compression. This demonstrates that Latent-SFT effectively compresses single explicit reasoning paths during the latent reasoning.

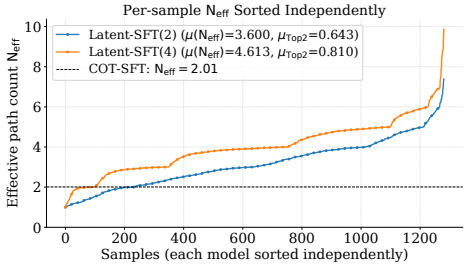


Figure 5: N_{eff} values of Latent-SFT on the Multi-Chain GSM8k-Aug dataset. The legend reports the mean values of both N_{eff} and the Top-2 Score. The dashed line indicates the empirical COT-SFT baseline under the original metric. Overall, Latent-SFT models across various compression ratios consistently exceed this baseline.

chain. The formal definition is provided in the Appendix J.2.

The experimental results are presented in Figure 4. Across models trained with different compression ratios r and varying K values, the ECR scores exceed 1 for almost all samples, quantitatively validating that latent reasoning compresses a single reasoning path. Moreover, comparing Figure 4(a) and (b), doubling the training compression ratio nearly doubles the ECR score during inference, further demonstrating the effectiveness of our training strategy.

Hypothesis 2. Latent Reasoning as the Superposition of Multiple Reasoning Chains. To examine whether latent reasoning corresponds to the superposition of multiple paths, we define the effective global parallelism (N_{eff}). A larger value reflects a higher degree of parallelism. In addition, we introduce the Top-2 Score, defined as the ratio of the probabilities of the two most likely paths. Formal definitions are provided in the Appendix J.3. To assess the parallel reasoning capability of latent models, we construct multiple alternative reasoning chains corresponding to a single explicit reasoning path within GSM8k-Aug dataset (Multi-Chain GSM8k-Aug dataset in Appendix I).

Figure 5 presents the analysis results. Across models with different compression ratios, most samples exceed the empirical COT-SFT baseline, indicating that latent reasoning supports more candidate paths than explicit SFT under the same metric. On average, evidence is distributed across roughly 3–4 reasoning paths rather than concentrated on a single path. The average Top-2 ratio exceeds 0.6, suggesting strong "two-way competition" and parallel reasoning. Furthermore, as the compression ratio increases, the degree of parallelism also rises. We provide a latent-to-latent matching analysis with a random baseline in Appendix J.4. A representative case study is provided in Figure 9 (Appendix J.5).

9 Conclusion

In this work, we introduce Latent-SFT, a unified framework that addresses latent reasoning at the token, chain, and learning levels through Latent-Vocab, Latent-Chain, and Latent-Optim. By doing so, we move latent reasoning beyond an unconstrained black box toward a structured process grounded in the vocabulary interface. Empirical evaluations across six mathematical benchmarks show that Latent-SFT consistently outperforms explicit SFT while reducing reasoning length by 2.7x–5.5x. Crucially, our analysis confirms that the learned latent process captures multiple parallel trajectories rather than merely compressing a single explicit chain. These findings suggest that latent reasoning can serve as an efficient and principled alternative to explicit chain-of-thought reasoning.

Limitations

Our study has several limitations. First, due to computational constraints, high-difficulty training uses a filtered Open-R1 subset with reasoning chains shorter than 4K tokens, which may underestimate the method's potential on longer competition-level problems. Second, our evaluation focuses primarily on mathematical reasoning; whether Latent-SFT generalizes to broader reasoning domains remains to be verified. Third, latent chains are constructed with fixed-length segmentation and a multi-stage training pipeline, which adds implementation complexity compared with standard CoT-SFT. Finally, the learned latent tokens are not directly human-interpretable, limiting fine-grained analysis of intermediate reasoning steps.

Ethical Considerations

Latent-SFT is intended to improve the efficiency of reasoning models, but it also introduces potential risks. Because latent reasoning steps are less directly interpretable than natural-language CoT, failures may be harder to diagnose or audit. In addition, our Multi-Chain analysis uses GPT-5 to generate candidate alternative reasoning chains; although these candidates are manually filtered, generated data may still reflect model biases or subtle reasoning errors. Finally, efficiency improvements in reasoning systems could lower the cost of both beneficial and harmful automated problem solving, so deployment should include task-appropriate monitoring and safeguards.

References

Natasha Butt, Ariel Kwiatkowski, Ismail Labiad, Julia Kempe, and Yann Ollivier. 2025. Soft tokens, hard truths. *CoRR*, abs/2509.19170.

Linbo Cao and Jinman Zhao. 2025. Pretraining on the test set is no longer all you need: A debate-driven approach to qa benchmarks. *Preprint*, arXiv:2507.17747.

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

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

Yuntian Deng, Yejin Choi, and Stuart M. Shieber. 2024. From explicit cot to implicit cot: Learning to internalize cot step by step. *CoRR*, abs/2405.14838.

Abhimanyu Dubey and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: program-aided language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

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

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

Hugging Face. 2025. Open r1: A fully open reproduction of deepseek-r1.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. 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, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2080–2094. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1743–1752. The Association for Computational Linguistics.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. 2025. CODI: compressing chain-of-thought into continuous space via self-distillation. *CoRR*, abs/2502.21074.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Jian Luan, and Ruihua Song. 2025. Think silently, think fast: Dynamic latent compression of LLM reasoning chains. *CoRR*, abs/2505.16552.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Zihao Wei, Jingcheng Deng, Liang Pang, Hanxing Ding, Huawei Shen, and Xueqi Cheng. 2025. Mlake: Multilingual knowledge editing benchmark for large language models. In *Proceedings of the 31st International Conference on Computational Linguistics, COLING 2025, Abu Dhabi, UAE, January 19-24, 2025*, pages 4457–4473. Association for Computational Linguistics.

Chünhung Wu, Jinliang Lu, Zixuan Ren, Gangqiang Hu, Zhi Wu, Dai Dai, and Hua Wu. 2025a. Llms are single-threaded reasoners: Demystifying the working mechanism of soft thinking. *Preprint*, arXiv:2508.03440.

Haoyi Wu, Zhihao Teng, and Kewei Tu. 2025b. Parallel continuous chain-of-thought with jacobi iteration. *CoRR*, abs/2506.18582.

Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2024. Search-in-the-chain: Interactively enhancing large language models with search for knowledge-intensive tasks. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, pages 1362–1373. ACM.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, and 1 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Boyi Zeng, Shixiang Song, Siyuan Huang, Yixuan Wang, He Li, Ziwei He, Xinbing Wang, Zhiyu Li, and Zhouhan Lin. 2025. Pretraining language models to ponder in continuous space. *CoRR*, abs/2505.20674.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, Shuohang Wang, Yelong Shen, and Xin Eric Wang. 2025. Soft thinking: Unlocking the reasoning potential of llms in continuous concept space. *CoRR*, abs/2505.15778.

Jinman Zhao and Xueyan Zhang. 2024. Large language model is not a (multilingual) compositional relation reasoner. In *First Conference on Language Modeling*.

Ruijie Zhu, Tianhao Peng, Tianhao Cheng, Xingwei Qu, Jinfa Huang, Dawei Zhu, Hao Wang, Kaiwen Xue, Xuanliang Zhang, Yong Shan, Tianle Cai, Taylor Kergan, Assel Kembay, Andrew Smith, Chenghua Lin, Binh Nguyen, Yuqi Pan, Yuhong Chou, Zefan Cai, and 14 others. 2025. A survey on latent reasoning. *CoRR*, abs/2507.06203.

Yufan Zhuang, Liyuan Liu, Chandan Singh, Jingbo Shang, and Jianfeng Gao. 2025. Text generation beyond discrete token sampling. *CoRR*, abs/2505.14827.

A Use of LLMs

We employ GPT-5 for two purposes. First, we use it to polish the language of our manuscript: we draft the text, ask GPT-5 to refine it, and then verify its logical and semantic consistency before incorporating the revision. Second, we use GPT-5 to generate preliminary candidate alternative reasoning chains for the Multi-Chain GSM8k-Aug analysis. These generated candidates are not used directly; they are manually inspected, filtered for format and logical correctness, and further filtered for diversity as described in Appendix I.

B Code and Data Availability

The models and datasets of our method are available at `https://anonymous.4open.science/r/Latent-SFT-CF17`.

C Distribution Gap after COCONUT Optimization

To examine whether latent-reasoning training reduces the mismatch between hidden states and token embeddings, we evaluate the released `agurung/coconut-qwen2.5-7b` checkpoint under the same measurement protocol as Section 4. As shown in Table 3, COCONUT optimization does not reduce the distributional discrepancy in this checkpoint; the hidden-state norm and FID both increase compared with the pretrained Qwen-2.5-7B model.

Table 3: Distribution gap between latent hidden states and the token-embedding space before and after COCONUT optimization.

Metric	Qwen-2.5-7B	coconut-qwen2.5-7b
hidden L2	110.27	134.52
embed L2	0.08	0.09
FID	14268.12	17421.55

D Structured Anisotropy of Vocab-Space

We analyze the singular value spectra of token embedding matrices across various LLMs to examine the geometry of vocab-space. Figure 6 shows that spectral energy is distributed unevenly across directions, indicating structured anisotropy rather than an unconstrained isotropic vector space.

E Training Computational Costs

We provide a detailed breakdown of the computational time required for both the **Latent-Chain**

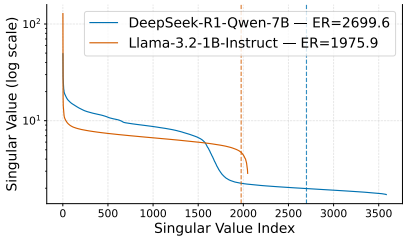


Figure 6: Decay curves of singular values for embedding matrices in various LLMs. The legend indicates the effective rank of each embedding matrix.

Construction and **Latent-Optim** phases. All models were trained on 8 × NVIDIA A100 GPUs.

The training process for Latent-SFT is divided into two primary steps. The first step, **Latent-Chain Construction**, involves three sub-phases: (1) latent-token encoder training, (2) LLM training, and (3) joint fine-tuning. The second step, **Latent-Optim**, involves training the LLM using the proposed stochastic Gumbel-Softmax strategy.

Table 4 summarizes the training duration for each model size. Despite the multi-stage nature of our framework, the total training cost remains manageable.

F Theoretical Analysis of Stochastic Latent Optimization

In this section, we analyze the implicit regularization effect of Gumbel-noise injection. Unlike standard Gaussian noise, Gumbel noise has a non-zero mean and fixed variance. We show that, combined with the translation invariance of Softmax, this induces a Hessian-based regularization term in expectation.

Theorem 1 (*Implicit Hessian Regularization via Gumbel-Softmax*). Let $\boldsymbol{\alpha} \in \Delta^{V-1}$ be a static target distribution in the simplex interior (i.e., $\alpha_i > 0$ and $\sum_i \alpha_i = 1$). Consider the stochastic target generated via Eq. (6): $\tilde{\mathbf{p}} = \sigma((\log \boldsymbol{\alpha} + \mathbf{g})/\tau)$, where $\sigma(\cdot)$ is Softmax, $\tau > 0$, and $\mathbf{g} \in \mathbb{R}^V$ has i.i.d. entries $g_i \sim \mathrm{Gumbel}(0, 1)$. Define the unperturbed baseline

$$\mathbf{p}_\tau \triangleq \sigma\big(\tau^{-1} \log \boldsymbol{\alpha}\big). \tag{8}$$

We analyze the expected loss in a neighborhood of $\mathbf{p}_\tau$ under the small-perturbation regime (defined below). Then, for a twice-differentiable loss $\mathcal{L}(\theta, \mathbf{p})$ that depends on $\mathbf{p}$ through the forward computation (hence generally non-linear in $\mathbf{p}$), we have:

$$\begin{aligned}\mathbb{E}_{\mathbf{g}}[\mathcal{L}(\theta, \tilde{\mathbf{p}})] = \mathcal{L}(\theta, \mathbf{p}_\tau) &+ \frac{\pi^2}{12\tau^2} \mathrm{Tr}\Big(\mathbf{J}_\sigma(\mathbf{p}_\tau)^\top \nabla^2_{\mathbf{p}} \mathcal{L}(\theta, \mathbf{p})\big|_{\mathbf{p}=\mathbf{p}_\tau} \\ &\qquad \mathbf{J}_\sigma(\mathbf{p}_\tau)\Big) \\ &+ \mathcal{O}\Big(\|\nabla_{\mathbf{p}} \mathcal{L}\| \, \mathbb{E}\|\boldsymbol{\delta}\|^2\Big) \\ &+ \mathcal{O}\Big(\mathbb{E}\|\boldsymbol{\delta}\|^3\Big),\end{aligned} \tag{9}$$

where $\mathbf{J}_\sigma(\mathbf{p}_\tau) = \mathrm{diag}(\mathbf{p}_\tau) - \mathbf{p}_\tau \mathbf{p}_\tau^\top$ is the Softmax Jacobian evaluated at $\mathbf{p}_\tau$, and $\boldsymbol{\delta}$ is the effective (zero-mean) perturbation in logit space defined in the proof. In particular, in our experiments with $\tau = 1$, $\mathbf{p}_\tau = \sigma(\log \boldsymbol{\alpha}) = \boldsymbol{\alpha}$, so the expansion is centered at the semantic target. This indicates that Gumbel noise induces a curvature penalty that favors flatter neighborhoods around the target distribution, with leading-order scaling $1/\tau^2$.

Proof.

1. Notation and mean removal via translation invariance. Let $\mathbf{l} = \log \boldsymbol{\alpha}$. The stochastic target is

$$\tilde{\mathbf{p}} = \sigma\left(\frac{\mathbf{l} + \mathbf{g}}{\tau}\right). \tag{10}$$

A standard Gumbel variable has mean $\mathbb{E}[g] = \gamma$ and variance $\mathrm{Var}(g) = \pi^2/6$. Decompose $\mathbf{g} = \boldsymbol{\epsilon} + \gamma \mathbf{1}$ where $\mathbb{E}[\boldsymbol{\epsilon}] = \mathbf{0}$ and $\mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top] = \frac{\pi^2}{6}\mathbf{I}$. Using Softmax translation invariance $\sigma(\mathbf{x} + c\mathbf{1}) = \sigma(\mathbf{x})$, we obtain

$$\tilde{\mathbf{p}} = \sigma\left(\frac{\mathbf{l} + \boldsymbol{\epsilon}}{\tau}\right) = \sigma\left(\frac{\mathbf{l}}{\tau} + \boldsymbol{\delta}\right), \qquad \boldsymbol{\delta} \triangleq \frac{\boldsymbol{\epsilon}}{\tau}. \tag{11}$$

Define the baseline $\mathbf{p}_\tau \triangleq \sigma(\mathbf{l}/\tau)$, so that $\tilde{\mathbf{p}} = \sigma(\mathbf{z}_\tau + \boldsymbol{\delta})$ with $\mathbf{z}_\tau = \mathbf{l}/\tau$ and $\tilde{\mathbf{p}}(\mathbf{0}) = \mathbf{p}_\tau$.

2. Second-order expansions. We assume a small-perturbation regime where $\|\boldsymbol{\delta}\|$ is sufficiently small such that second-order Taylor expansions are valid and the remainders are controlled by $\mathcal{O}(\|\boldsymbol{\delta}\|^3)$ in expectation.

Softmax expansion. Let $\mathcal{H}_\sigma$ denote the (third-order) Hessian tensor of Softmax with respect to logits. Using tensor contraction notation $\mathcal{H}_\sigma[\boldsymbol{\delta}, \boldsymbol{\delta}] \in \mathbb{R}^V$,

$$\tilde{\mathbf{p}} - \mathbf{p}_\tau = \mathbf{J}_\sigma(\mathbf{p}_\tau)\, \boldsymbol{\delta} + \frac{1}{2}\, \mathcal{H}_\sigma[\boldsymbol{\delta}, \boldsymbol{\delta}] + \mathcal{O}(\|\boldsymbol{\delta}\|^3). \tag{12}$$

Loss expansion. Expanding $\mathcal{L}(\theta, \mathbf{p})$ around $\mathbf{p} =$

Table 4: Training time breakdown (in hours) for Latent-SFT across different model scales.

Base Model	**Latent-Chain Construction (h)**				**Latent-Optim (h)**	**Total (h)**
	Latent Enc.	*LLM*	*Joint*	**Subtotal**		
LLaMA-3.2-1B-Instruct	4.0	4.0	4.5	12.5	12.0	24.5
LLaMA-3.2-3B-Instruct	6.5	6.5	8.0	21.0	22.0	43.0
DeepSeek-Distill-Qwen-7B	12.0	12.0	15.0	39.0	50.0	89.0

$\mathbf{p}_\tau$,

$$\mathcal{L}(\theta, \tilde{\mathbf{p}}) = \mathcal{L}(\theta, \mathbf{p}_\tau) + (\tilde{\mathbf{p}} - \mathbf{p}_\tau)^\top \nabla_{\mathbf{p}} \mathcal{L}(\theta, \mathbf{p})\big|_{\mathbf{p}=\mathbf{p}_\tau} + \frac{1}{2}(\tilde{\mathbf{p}} - \mathbf{p}_\tau)^\top \nabla^2_{\mathbf{p}} \mathcal{L}(\theta, \mathbf{p})\big|_{\mathbf{p}=\mathbf{p}_\tau} (\tilde{\mathbf{p}} - \mathbf{p}_\tau) + \mathcal{O}(\|\tilde{\mathbf{p}} - \mathbf{p}_\tau\|^3). \tag{13}$$

Substituting the first-order relation $\tilde{\mathbf{p}} - \mathbf{p}_\tau = \mathbf{J}_\sigma \boldsymbol{\delta} + \mathcal{O}(\|\boldsymbol{\delta}\|^2)$ into the quadratic term yields

$$\mathcal{L}(\theta, \tilde{\mathbf{p}}) = \mathcal{L}(\theta, \mathbf{p}_\tau) + \left(\mathbf{J}_\sigma \boldsymbol{\delta} + \frac{1}{2}\mathcal{H}_\sigma[\boldsymbol{\delta}, \boldsymbol{\delta}]\right)^\top \nabla_{\mathbf{p}} \mathcal{L}\big|_{\mathbf{p}=\mathbf{p}_\tau} + \frac{1}{2}\boldsymbol{\delta}^\top \mathbf{J}_\sigma^\top \nabla^2_{\mathbf{p}} \mathcal{L}\big|_{\mathbf{p}=\mathbf{p}_\tau} \mathbf{J}_\sigma\, \boldsymbol{\delta} + \mathcal{O}(\|\boldsymbol{\delta}\|^3), \tag{14}$$

where $\mathbf{J}_\sigma = \mathbf{J}_\sigma(\mathbf{p}_\tau)$ for brevity.

3. Taking expectations. Since $\mathbb{E}[\boldsymbol{\delta}] = \mathbf{0}$, the linear-in-$\boldsymbol{\delta}$ term vanishes: $\mathbb{E}[\mathbf{J}_\sigma \boldsymbol{\delta}] = \mathbf{0}$. The remaining first-order bias arises from Softmax nonlinearity:

$$\mathbb{E}\Big[\big(\tfrac{1}{2}\mathcal{H}_\sigma[\boldsymbol{\delta}, \boldsymbol{\delta}]\big)^\top \nabla_{\mathbf{p}} \mathcal{L}\big|_{\mathbf{p}=\mathbf{p}_\tau}\Big] = \mathcal{O}\Big(\|\nabla_{\mathbf{p}} \mathcal{L}\|\, \mathbb{E}\|\boldsymbol{\delta}\|^2\Big), \tag{15}$$

which becomes negligible near a stationary point where $\|\nabla_{\mathbf{p}} \mathcal{L}\| \approx 0$.

For the quadratic term, let $\mathbf{H}_\tau = \nabla^2_{\mathbf{p}} \mathcal{L}\big|_{\mathbf{p}=\mathbf{p}_\tau}$ and $\mathbf{A}_\tau = \mathbf{J}_\sigma^\top \mathbf{H}_\tau \mathbf{J}_\sigma$. Then

$$\mathbb{E}\left[\frac{1}{2}\boldsymbol{\delta}^\top \mathbf{A}_\tau \boldsymbol{\delta}\right] = \frac{1}{2}\operatorname{Tr}\Big(\mathbf{A}_\tau \mathbb{E}[\boldsymbol{\delta}\boldsymbol{\delta}^\top]\Big). \tag{16}$$

Using $\mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top] = \frac{\pi^2}{6}\mathbf{I}$ and $\boldsymbol{\delta} = \boldsymbol{\epsilon}/\tau$ gives $\mathbb{E}[\boldsymbol{\delta}\boldsymbol{\delta}^\top] = \frac{\pi^2}{6\tau^2}\mathbf{I}$, hence

$$\mathbb{E}\left[\frac{1}{2}\boldsymbol{\delta}^\top \mathbf{A}_\tau \boldsymbol{\delta}\right] = \frac{\pi^2}{12\tau^2}\operatorname{Tr}(\mathbf{A}_\tau). \tag{17}$$

Collecting terms yields Eq. (9), with the remainder $\mathcal{O}(\mathbb{E}\|\boldsymbol{\delta}\|^3)$ from Taylor truncation. □

G Latent-Chain Construction Training

The strict constraints imposed by LTIM and LT-SuM can destabilize naive SFT optimization. To address this, we adopt an EM-style alternating optimization schedule. Let $\theta = \{\theta_{\text{enc}}, \theta_{\text{llm}}\}$ represent the trainable parameters, where θ_{enc} denotes the latent-token encoder initialized from the base LLM and θ_{llm} denotes the target LLM. The training proceeds in three phases:

1. **Latent-Token Encoder Optimization:** Freezing the LLM backbone, we update θ_{enc} to minimize reconstruction loss: $\theta_{\text{enc}}^{(t+1)} \leftarrow \arg\min_{\theta_{\text{enc}}} \mathcal{L}_{\text{sup}}(\theta_{\text{enc}}, \theta_{\text{llm}}^{(t)})$.
2. **LLM Optimization:** Freezing the latent-token encoder, we update the LLM backbone to adapt to the latent representations: $\theta_{\text{llm}}^{(t+1)} \leftarrow \arg\min_{\theta_{\text{llm}}} \mathcal{L}_{\text{sup}}(\theta_{\text{enc}}^{(t+1)}, \theta_{\text{llm}})$.
3. **Joint Fine-tuning:** Finally, we jointly optimize both components to ensure global convergence.

This progressive strategy stabilizes the training dynamics, allowing the model to gradually adapt to the distributional shift introduced by the latent tokens.

G.1 Fixed-Length Segmentation

In our experiments, $\text{Seg}(\cdot)$ uses a fixed compression ratio r. Given a tokenized explicit reasoning chain $C = (c_1, \ldots, c_L)$, we set

$$N = \left\lceil \frac{L}{r} \right\rceil, \tag{18}$$

and define each segment as

$$S_i = \big(c_{(i-1)r+1}, \ldots, c_{\min(ir,L)}\big), \quad i = 1, \ldots, N. \tag{19}$$

We then insert one latent placeholder L_i immediately after each segment S_i, yielding the augmented chain $[S_1, L_1, S_2, L_2, \ldots, S_N, L_N]$. The label of

each placeholder token is set to -100 so that it is ignored by the standard token-level cross-entropy loss.

H Implementation Details

Evaluation Metrics and Protocol. Our evaluation measures both accuracy (Pass@N) and efficiency (# L, the number of tokens in the reasoning chain). For the low-difficulty datasets, we conducted five runs with different random seeds. For the high-difficulty datasets, Math500 was evaluated four times, while AIME24 was evaluated 64 times. We report Pass@1 and # L as the mean $\pm$ 95% confidence interval (CI). To jointly assess accuracy and efficiency, we also report their ratio, Pass@1/# L.

Training and Inference Setup. Since Latent-SFT requires weighting with respect to the vocabulary matrix, we freeze the vocabulary parameters during training. All models are trained using LoRA with rank 64. Following COCONUT, we initialize all methods from the COT-SFT model to accelerate convergence. In practice, we find that a fixed-length segmentation function performs better than a semantic-based segmentation. Therefore, all results reported in this paper are based on fixed-length segmentation functions. For LLaMA-3.2-1B-Instruct and LLaMA-3.2-3B-Instruct, we set the learning rate to 1×10^{-4} for Latent-Chain generation and 4×10^{-4} for Latent-Optim. For DeepSeek-Distill-Qwen-7B, the learning rates are 3×10^{-5} (Latent-Chain) and 1×10^{-5} (Latent-Optim). Following (DeepSeek-AI, 2025), we use a decoding temperature of 0.6 and top-p of 0.95.

During evaluation, low-difficulty mathematical reasoning datasets were inferred using the Transformers library with a maximum sequence length of 128. High-difficulty datasets were evaluated with the improved SGlang library, supporting a maximum token length of 32,768. All training and inference experiments were conducted on 8×A100 GPUs.

I Construction of the Multi-Chain GSM8k-Aug Dataset

We carefully designed prompts and employed GPT-5 to generate preliminary candidate reasoning chains. All generated candidates were manually inspected by the authors, and candidates that failed to follow the required reasoning format or contained incorrect logic were discarded. To further maximize diversity, we retained only reasoning chains with pairwise similarity below 0.9 based on edit distance. An example prompt call is illustrated in Figure 8. Among the 1,319 data points, 1,310 contained multiple reasoning paths after filtering. The resulting Multi-Chain data are released with the anonymous repository in Appendix B.

J Analytical Metrics

J.1 Notation and Preliminaries

Let $\mathcal{V}$ be the vocabulary, $|\mathcal{V}| = V$. For a given problem, fix a reference explicit chain (from an explicit CoT solution)

$$\boldsymbol{x} = (x_1, \ldots, x_{L_{\text{exp}}}), \quad x_i \in \mathcal{V}. \tag{20}$$

A latent model generates a latent reasoning chain consisting of T latent steps, each with a vocabulary distribution

$$p_t \in \Delta^{V-1} \quad (t = 1, \ldots, T), \tag{21}$$

where Δ^{V-1} is the probability simplex. To quantify the extent of explicit information captured by each latent step (whether from a single chain or multiple chains), we introduce:

Assumption 1. During training, if each latent token is constrained to receive information from a fixed number of explicit tokens, then at inference time the amount of explicit information (in terms of token granularity) captured by each latent step is uniform.

This assumption is reasonable, as the model is expected to behave consistently during training and inference. So we align the two sequences by a ratio $r \in \mathbb{N}$: every r explicit tokens are grouped to one latent step. Define the number of aligned steps

$$T' = \min\big(T, \lceil L_{\text{exp}}/r \rceil\big), \tag{22}$$

and the explicit token set aligned to latent step t:

$$\begin{aligned} S_t &= \left\{x_{(t-1)r+1}, \ \ldots, \ x_{\min(tr, L_{\text{exp}})}\right\}, \\ |S_t| &\le r. \end{aligned} \tag{23}$$

Let $\mathcal{T}_K(p_t) \subset \mathcal{V}$ be the set of the K highest-probability tokens under p_t.

J.2 ECR@K (Effective Compression Rate)

To verify whether latent reasoning corresponds to the compression of a single chain, we examine, *on average, how many aligned explicit tokens from the*

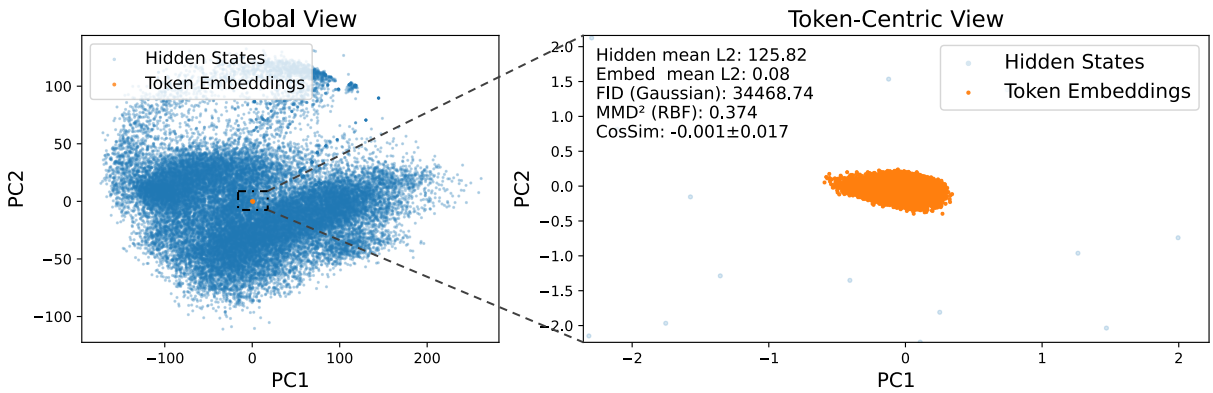


Figure 7: Visualization of last-layer hidden states and token embeddings of the Deepseek-Distill-Qwen-7B model on the Math500 dataset.

Instruction:

"Given a GSM8k problem, solution, and answer, identify alternative valid reasoning paths. Enclose each path within \\boxed{}. The paths must not contain any natural language, and should follow the same style as the provided example. Ensure the length of the solution and the number of reasoning steps <<>> remain consistent with the standard solution. Each reasoning path must strictly adhere to the problem's logical structure—do not generate unrelated or fabricated paths. If no alternative reasoning path exists, output \\boxed{False}.

{"problem": "Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?", "solution": "<<16-3-4=9>> <<9*2=18>>", "answer": "18"}"

Response:

"\boxed{<<<<16-(3+4)=9>> <<9*2=18>>} ✗ Rejected (edit-distance criterion against the solution violated).
\boxed{<<3+4=7>> <<(16-7)*2=18>>}
\boxed{<<16-3=13>> <<(13-4)*2=18>>}
\boxed{<<16-4=12>> <<(12-3)*2=18>>} ✗ Rejected (edit-distance criterion against the previous item violated).
\boxed{<<2*16=32>> <<32-2*3-2*4=18>>}"

Figure 8: Illustration of the Multi-Chain dataset construction pipeline. We prompt GPT-5 to generate alternative valid reasoning paths for a given GSM8k problem. The response section demonstrates the diversity filtering mechanism: candidate paths are evaluated based on edit distance, and those exhibiting high similarity to the standard solution or previously accepted paths (marked with $\times$) are rejected to ensure significant reasoning diversity.

reference chain appear within the Top-K probabilities at each latent step. Thus, we define

$$\text{ECR@K} \;=\; \frac{1}{T'}\sum_{t=1}^{T'}\big|S_t \cap \mathcal{T}_K(p_t)\big| \;\in\; [0, r]. \tag{24}$$

A value greater than 1 indicates genuine compression of a single reasoning chain, meaning that each latent step, on average, covers more than one token from the reference explicit chain.

J.3 $\mathrm{N}_{\mathrm{eff}}$ and Top-2 Score

Given M candidate explicit chains $\{\boldsymbol{x}^{(m)}\}_{m=1}^{M}$ (with lengths L_m), we align each chain to the T latent steps using the same ratio r as above. Let the explicit tokens aligned to step t on chain m be $S_{m,t} \subseteq \mathcal{V}$ (at most r tokens), and let $\mathcal{T}_K(p_t)$ be the Top-K tokens under p_t. Define the per-step *mass* of chain m as

$$\mathrm{mass}_{m,t}^{(K)} \;=\; \sum_{x \in S_{m,t} \cap \mathcal{T}_K(p_t)} p_t[x], \tag{25}$$

and aggregate it across steps to obtain a sequence-level evidence score

$$\mathrm{score}_m \;=\; \frac{1}{T'}\sum_{t=1}^{T'} \log\big(\mathrm{mass}_{m,t}^{(K)} + \varepsilon\big), \tag{26}$$

where $T' = \min\big(T, \lceil L_m/r \rceil\big)$, $\varepsilon > 0$ is a small constant, and K controls robustness (setting $K = V$ recovers the full-mass version). We convert $\{\mathrm{score}_m\}$ to a posterior over paths via a tempered softmax

$$P_m \;=\; \frac{\exp(\mathrm{score}_m/\tau)}{\sum_{j=1}^{M}\exp(\mathrm{score}_j/\tau)} \quad (\tau = 1). \tag{27}$$

The **effective number of supported paths** is

$$\mathrm{N_{eff}} = \exp\Big(-\sum_{m=1}^{M} P_m \log P_m\Big), \tag{28}$$

and the **Top-2 score** is the ratio between the two largest posteriors,

$$\text{Top-2} = \frac{P_{(2)}}{P_{(1)}} \quad \text{with} \quad P_{(1)} \geq P_{(2)}. \tag{29}$$

Intuitively, a larger score_m means the model consistently places high probability on chain m along the sequence; a larger $\mathrm{N_{eff}}$ and higher Top-2 indicate stronger parallel support across multiple chains rather than collapse to a single path.

When only a single path exists in parallel, $\mathrm{N_{eff}} = 1$. However, different paths may partially overlap in prefixes and syntax tokens, so we use COT-SFT as an empirical baseline instead of a fixed heuristic threshold. On the same Multi-Chain GSM8k-Aug data, COT-SFT obtains $\mathrm{N_{eff}} = 2.01$ with the original metric and $\mathrm{N_{eff}} = 1.66$ after excluding syntax tokens « and ». In addition, to mitigate the influence of tail noise, we set $K = 100$.

J.4 Latent-to-Latent Matching with Random Baseline

The original $\mathrm{N_{eff}}$ metric measures support for multiple chains through token-level overlap. While intuitive, it can be affected by fixed positional alignment, shared syntax tokens, or the normalization in the posterior over paths. To provide a stricter check, we compare inferred latent states directly with the latent states induced by each candidate trajectory, and calibrate the result with a random no-repeat matching baseline.

For each candidate trajectory m, we first encode its explicit segments into reference latent states $\{\mathbf{z}_{m,s}\}_{s=1}^{T_m}$ using the same Latent-Chain construction process. Given an inferred latent chain $\{\hat{\mathbf{z}}_t\}_{t=1}^{T}$, define $T'_m = \min(T, T_m)$ and compute the latent-to-latent distance matrix

$$D_m[t, s] = \mathrm{dist}(\hat{\mathbf{z}}_t, \mathbf{z}_{m,s}), \tag{30}$$

where $\mathrm{dist}(\cdot,\cdot)$ is the latent-space distance. We then find the best no-repeat alignment

$$\pi^\star_m = \arg\min_{\pi \in \Pi_m} \frac{1}{T'_m} \sum_{t=1}^{T'_m} D_m[t, \pi(t)], \tag{31}$$

where Π_m is the set of injective mappings from inferred latent steps to reference segments. The trajectory-level matching score is

$$a_m = -\frac{1}{T'_m} \sum_{t=1}^{T'_m} D_m[t, \pi^\star_m(t)]. \tag{32}$$

We convert $\{a_m\}_{m=1}^{M}$ into a posterior over candidate trajectories and compute the effective number of matched paths:

$$P_m^{\mathrm{match}} = \frac{\exp(a_m/\tau)}{\sum_{j=1}^{M} \exp(a_j/\tau)},$$
$$N_{\mathrm{eff}} = \exp\left(-\sum_{m=1}^{M} P_m^{\mathrm{match}} \log P_m^{\mathrm{match}}\right). \tag{33}$$

For calibration, we sample random no-repeat mappings $\pi^{\mathrm{rand}}_{m,b} \in \Pi_m$ and compute the same quantity, yielding $N_{\mathrm{eff,naive}}$. If a high $\mathrm{N_{eff}}$ only results from posterior normalization or random path overlap, then $N_{\mathrm{eff}}/N_{\mathrm{eff,naive}}$ should be close to 1. A value greater than 1 indicates that the inferred latent chain matches valid alternative trajectories better than random alignment.

Table 5: Latent-to-latent matching analysis with a random no-repeat baseline on Multi-Chain GSM8k-Aug.

Model	$\mu(N_{\mathrm{eff}})$	$\mu(N_{\mathrm{eff,naive}})$	$\mu(\frac{N_{\mathrm{eff}}}{N_{\mathrm{eff,naive}}})$
Latent-SFT(2)	2.97	1.86	1.59
Latent-SFT(4)	3.42	2.51	1.36

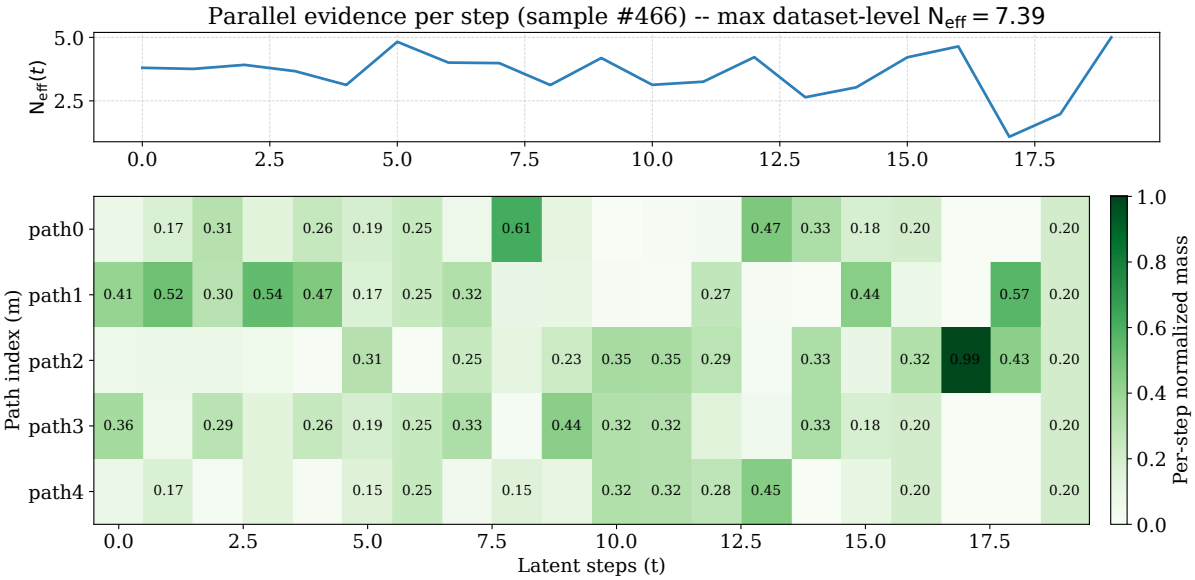
We also compare the original $\mathrm{N_{eff}}$ metric with COT-SFT under the same candidate set. As shown in Table 6, Latent-SFT remains substantially higher than COT-SFT, even after excluding syntax tokens « and » from the counted vocabulary.

Table 6: Original $\mathrm{N_{eff}}$ metric compared with COT-SFT on Multi-Chain GSM8k-Aug.

Model	Original $\mu(N_{\mathrm{eff}})$	w/o syntax tokens
Latent-SFT(2)	3.60	3.10
Latent-SFT(4)	4.61	3.98
COT-SFT	2.01	1.66

J.5 $\mathrm{N_{eff}}$ Case Study

Figure 9 illustrates a representative example with the maximum $\mathrm{N_{eff}}$ value for Latent-SFT(2) on the Multi-Chain GSM8k-Aug dataset.



Question: Becky bought 20 apples for 45 cents each and received a $1 discount. Kelly bought 20 apples for 50 cents each and received a 10 percent discount. How much more did Kelly pay than Becky?
Path0: <<20*0.45=9>> <<9-1=8>> <<20*0.5=10>> <<(10*10)/100=1>> <<10-1=9>> <<9-8=1>>
Path1: <<45*20=900>> <<900-100=800>> <<50*20=1000>> <<1000-100=900>> <<900-800=100>> <<100/100=1>>
Path2: <<0.50-0.45=0.05>> <<0.05*20=1>> <<20*0.50=10>> <<(10*10)/100=1>> <<1-1=0>> <<1+0=1>>
Path3: <<(45/100)*20=9>> <<9-1=8>> <<(50/100)*20=10>> <<10*(1/10)=1>> <<10-1=9>> <<9-8=1>>
Path4: <<20*0.45=9>> <<9-1=8>> <<20*0.50=10>> <<10*9=90>> <<90/10=9>> <<9-8=1>>

Figure 9: An example of the maximum N_{eff} value for the Latent-SFT(2) model on the Multi-Chain GSM8k-Aug dataset. At each step, the N_{eff} value exceeds the empirical COT-SFT baseline. Path0 denotes the original reasoning chain, while the remaining paths are additional candidate chains. The heatmap illustrates the fraction of paths supported at each latent step, revealing evidence of parallel reasoning.

Question:
"Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?"

Reference Answer:
"<<16-3-4=9>> <<9*2=18>>\\boxed{18}",

Latent-SFT(2) Response:
"47 <<000=>>9218 <<18\\boxed{18}"

Top-1 Tokens in Latent Reasoning

Figure 10: Reasoning example of Latent-SFT(2) on the GSM8k-Aug test set. The model generates a sequence of latent tokens (annotated as "Top-1 Tokens") instead of explicit natural language steps. While these tokens appear as unintelligible numeric combinations to humans, they effectively encode the necessary logical computations, allowing the model to derive the correct answer 18. Notably, this latent process reduces the reasoning chain length by nearly 50% compared to the reference solution, demonstrating high computational efficiency.

Algorithm 1 Latent-SFT: Latent-Chain Construction, Latent Optimization, and Inference

Require: Question Q, explicit reasoning chain C, final answer A, segmentation function $\mathrm{Seg}(\cdot)$, token embedding matrix $\mathbf{E}$, LM head $\mathbf{W}$, latent-token encoder θ_{enc}, target LLM θ_{llm}, maximum latent length $T_{\max}$.

Ensure: A target LLM that performs reasoning through latent superposition tokens.

Phase A: Construct Latent-Chain Targets

Split the explicit chain into N segments:

$$(S_1, \ldots, S_N) \leftarrow \mathrm{Seg}(C).$$

Insert one latent placeholder after each segment:

$$\bar{X} = [Q, \langle\mathrm{think}\rangle, S_1, L_1, S_2, L_2, \ldots, S_N, L_N, \langle/\mathrm{think}\rangle, A].$$

for $i = 1, \ldots, N$ **do**

Latent-token induction. Run the latent-token encoder on $\bar{X}$ under the Latent Token Induction Mask (LTIM), where placeholder L_i attends only to

$$[Q, \langle\mathrm{think}\rangle, S_1, \ldots, S_i].$$

Future segments and latent placeholders are masked.

Extract the hidden state at L_i:

$$h_i \leftarrow \theta_{\mathrm{enc}}(L_i; \mathrm{LTIM}).$$

Vocab-space alignment. Project h_i to the vocabulary simplex and reconstruct the superposition embedding:

$$\alpha_i = \mathrm{softmax}(\mathbf{W}^\top h_i / \tau_{\mathrm{enc}}), \qquad z_i = \mathbf{E}\alpha_i.$$

end for

Semantic sufficiency supervision. For each i, define the future explicit target

$$Y_i = [S_{i+1}, \ldots, S_N, \langle/\mathrm{think}\rangle, A],$$

with $Y_N = [\langle/\mathrm{think}\rangle, A]$.

Optimize the construction process under the Latent Token Supervision Mask (LTSuM): when predicting tokens in Y_i, the model uses the latent prefix

$$[Q, \langle\mathrm{think}\rangle, z_1, \ldots, z_i]$$

and the causal prefix inside Y_i, while past explicit segments $S_{\leq i}$ and future latent tokens $z_{>i}$ are masked.

Train Phase A progressively:

$$\theta_{\mathrm{enc}} \to \theta_{\mathrm{llm}} \to (\theta_{\mathrm{enc}}, \theta_{\mathrm{llm}}),$$

i.e., first optimize the latent-token encoder, then the target LLM, and finally jointly fine-tune both modules.

Store the constructed latent-chain target

$$\mathcal{T}(Q, C, A) = \{(\alpha_i, z_i)\}_{i=1}^{N}.$$

Phase B: Train the Target LLM with Latent-Optim

Remove all explicit intermediate reasoning segments and form the latent training sequence:

$$X_{\mathrm{lat}} = [Q, \langle\mathrm{think}\rangle, \tilde{z}_1, \ldots, \tilde{z}_N, \langle/\mathrm{think}\rangle, A].$$

for each training update and each latent position i **do**

Sample Gumbel noise:

$$g_i \sim \mathrm{Gumbel}(0, 1)^V.$$

Construct the stochastic latent distribution:

$$\tilde{p}_i = \mathrm{softmax}\left(\frac{\log \alpha_i + \lambda g_i}{\tau_g}\right).$$

Use the same stochastic distribution both as the KL target and as the teacher-forced latent input:

$$\tilde{z}_i = \mathbf{E}\tilde{p}_i.$$

end for

Let $q_i = \mathrm{softmax}(\mathbf{W}^\top h_i^{\mathrm{llm}})$ be the target LLM prediction at latent position i. Optimize

$$\mathcal{L}_{\mathrm{auto}} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{g_i} \mathrm{KL}(\tilde{p}_i \,\|\, q_i) + \frac{1}{|S_{\mathrm{exp}}|} \sum_{t \in S_{\mathrm{exp}}} -\log q_t[y_t],$$

where S_{exp} denotes the ordinary discrete tokens after the latent chain, including $\langle/\mathrm{think}\rangle$ and answer tokens.

Inference

Discard the latent-chain construction encoder θ_{enc}. Initialize the reasoning prefix:

$$P_0 = [Q, \langle\mathrm{think}\rangle].$$

Set $t \leftarrow 1$.

while $t \leq T_{\max}$ **do**

The target LLM produces a hidden state h_t from the current prefix P_{t-1}.

Convert the hidden state into a vocabulary distribution:

$$\alpha_t = \mathrm{softmax}(\mathbf{W}^\top h_t).$$

Decode the next discrete candidate token:

$$y_t \leftarrow \mathrm{Decode}(\alpha_t).$$

if $y_t = \langle/\mathrm{think}\rangle$ **then**

Append the discrete stop token:

$$P_t = [P_{t-1}, \langle/\mathrm{think}\rangle].$$

break

else

Construct and append the latent superposition token:

$$z_t = \mathbf{E}\alpha_t, \qquad P_t = [P_{t-1}, z_t].$$

$t \leftarrow t + 1$.

end if

end while

Decode the final answer autoregressively in the standard token space from the resulting prefix.