Why Struggle with Continuous Latents? Interpretable Discrete Latent Reasoning via Rendered Compression

Shuochen Chang[1*] **Qingyang Liu**[1*] **Shaobo Wang**[1] **Bingjie Gao**[1] **Qianli Ma**[1]
Haonan Zhao[1] **Yibo Miao**[1] **Yulin Sun**[1] **Zelin Peng**[1] **Jiangtong Li**[2†] **Li Niu**[1†]
[1]Shanghai Jiao Tong University [2]Tongji University
[*]Equal contribution. [†]Corresponding authors.
{csc1332741686, narumimaria, ustcnewly}@sjtu.edu.cn
jiangtongli@tongji.edu.cn

Abstract

Large language models achieve high reasoning performance via explicit chain-of-thought and reinforcement learning, but require long output sequences and extended inference time. Latent reasoning reduces this cost by shifting computation into a latent space; however, continuous latent methods are hard to train, suffering from unstable and uninterpretable reasoning trajectories. We argue these issues stem from a misalignment between continuous-space reasoning and discrete symbolic supervision, as continuous states lack explicit anchors for step-by-step alignment. To resolve this, we propose **Discrete Latent Reasoning (DLR)**, the first method that converts continuous latent states into explicit discrete tokens. Inspired by render-based compression, we render textual chains of thought into images, extract visual features, and construct a discrete latent vocabulary via clustering-based fine-tuning. Expanding the vocabulary and output head enables standard autoregressive modeling over both natural language and latent tokens, supporting pretraining alignment, SFT, and RL. Experiments on five reasoning benchmarks and two model series (Qwen3-VL and LLaMA-3) confirm that **DLR** outperforms prior latent reasoning baselines with up to **20× compression**. Furthermore, the learned latent trajectories retain an interpretable semantic structure. Overall, discrete latent tokens provide a controllable and interpretable basis for efficient latent reasoning. Code is available at `github.com/Miraclecsc/Discrete-Latent-Reasoning`.

1 Introduction

Large language models (LLMs) have achieved high reasoning performance via chain-of-thought (CoT) and reinforcement learning (RL) [51, 19, 15, 39]. However, they often require large token budgets and extended inference time to solve difficult problems. Latent reasoning offers an efficient alternative by shifting intermediate computation into a high-bandwidth latent space, preserving uncertainty structures while reducing verbalization costs. This approach is supported by recent visual intermediate representations [50, 46, 6, 25, 48], which indicate that reasoning can be effectively compressed in image space.

Recent analyses of representative latent systems [26, 61, 60] reveal that continuous latent tokens are highly susceptible to representation collapse, often remaining overly similar and yielding limited impact on the final answer [59, 24, 44, 34]. These issues expose the core vulnerabilities of continuous latent models, which primarily suffer from three critical limitations. First, these models introduce a major **training bottleneck**. During supervised fine-tuning (SFT), continuous latent states lack explicit discrete anchors analogous to text tokens, making teacher forcing difficult and frequently requiring sequential unrolling of latent trajectories, which reduces efficiency and compatibility

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

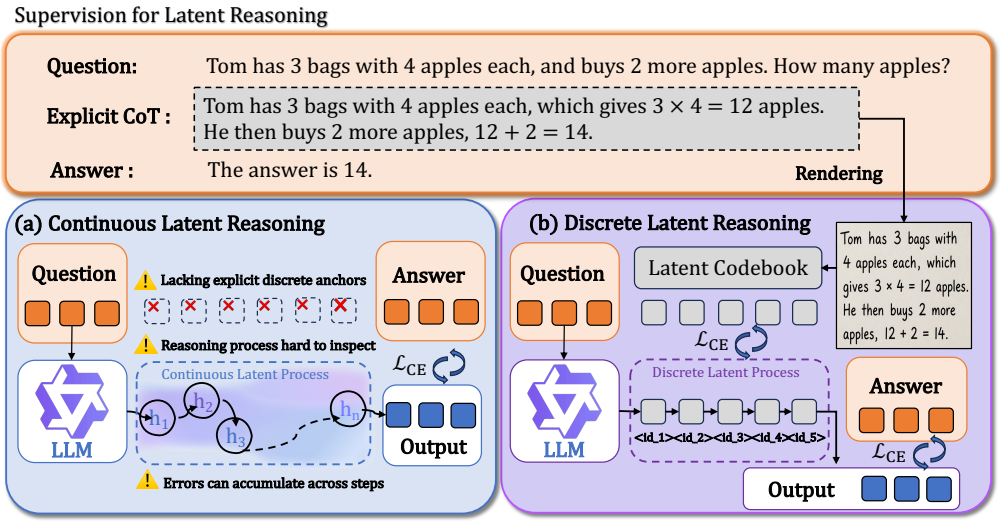


Figure 1: Continuous latent reasoning suffers from a mismatch between its intermediate states and discrete symbolic supervision, resulting in a training bottleneck, unstable reasoning dynamics, and poor interpretability. Our approach resolves this by introducing discrete latent tokens as anchors within the reasoning process, enabling standard autoregressive training and controllable latent reasoning.

with existing SFT and RL methods [17, 55]. During RL, the absence of explicit step boundaries complicates the assignment of process-level rewards and disrupts stable credit assignment over latent transitions [27, 53]. Second, they frequently display **unstable reasoning dynamics**. Since the latent channel is continuous and only weakly constrained, local errors can accumulate across steps, causing instability, representation collapse, or deviation from decision-relevant evidence [64, 52]. Moreover, unlike explicit textual reasoning, latent states lack strong language modeling priors from natural language tokens, limiting their stability and compositionality [10, 8]. Third, latent reasoning is largely opaque and suffers from **poor interpretability**, making the reasoning process hard to inspect, verify, or correct in reliability-critical settings [5, 22, 35].

The above limitations stem from a **misalignment between continuous-space reasoning and discrete symbolic supervision**. Unlike standard autoregressive methods built upon discrete vocabularies [3, 32], continuous latent reasoning lacks explicit anchors, making intermediate states difficult to supervise, inspect, or correct [10, 8]. To retain computational efficiency while enabling stable and interpretable training, we propose resolving this by **discretizing latent states**, *i.e.*, transforming implicit reasoning steps into explicit latent tokens directly compatible with standard language modeling.

In this work, we propose **Discrete Latent Reasoning (DLR)** (Fig. 1), a novel method that converts continuous latent states into discrete latent tokens for latent reasoning. Inspired by the render-compression strategy of DeepSeek-OCR [50], we design a **stochastic latent codebook construction module**, which renders textual chains of thought into images and leverages optical compression to extract dense, semantically structured visual features for latent tokenization. We then introduce a stochastic clustering-based fine-tuning mechanism to construct a discrete latent vocabulary, where each token corresponds to an interpretable semantic latent unit. To enable autoregressive modeling over these units, we propose an **augmented latent language model training pipeline**, which expands both the vocabulary and the output head of LLMs, allowing them to process and generate sequences in an augmented space comprising both natural language and latent tokens. The training proceeds in three stages: ***(i) latent-text alignment*** to synchronize the latent embeddings in the LLM space; ***(ii) latent SFT*** to teach the model to generate correct latent trajectories; and ***(iii) latent RL*** to optimize the latent-token policy via outcome and process rewards. This allows latent reasoning to be optimized via next-token prediction, supports process-level rewards over discrete steps, reduces error accumulation through structured anchors, and improves interpretability via an auxiliary decoder.

To validate the effectiveness of our approach, we fine-tune both large language models (LLaMA3) and large multimodal models (Qwen3-VL) on the GSM8K-Aug-train [7] split. Compared to previous latent reasoning baselines such as RoT [46], ReGuLaR [43], and CODI [38], our method achieves consistent accuracy improvements across GSM8K [7], GSM-Hard [13], SVAMP [33], MultiArith [37], and MATH-500 [18]. Moreover, we conduct experiments on the scalability of **DLR** and the interpretability of the latent trajectories, finding that the model can scale from 1B to 8B LLMs while retaining a meaningful degree of interpretability. Our contributions are summarized as:

- We propose a **Stochastic Latent Codebook Construction** pipeline that discretizes continuous reasoning states into semantically recoverable latent tokens through rendered CoT, providing interpretable latent representations with up to **20× compression**.
- We design **an Augmented Latent LM Training Pipeline**, which enables standard next-token prediction during SFT and supports process-reward RL. This design maintains semantic fidelity while significantly improving training stability.
- Together, we present **Discrete Latent Reasoning (DLR)**, the first discrete latent reasoning framework, and show its superiority over latent reasoning baselines on five reasoning benchmarks with improved trajectory interpretability and scalability.

2 Related Works

2.1 Continuous Latent Reasoning

Latent reasoning improves efficiency by replacing verbose text with dense continuous states. For instance, Coconut [17] introduces autoregressive reasoning in latent space by utilizing final hidden states. Subsequent works advance this approach: CODI [38] aligns trajectories via self-distillation, CoLaR [40] uses token-guided dynamic compression, and SIM-CoT [52] incorporates supervised decoding signals. Additional research investigates KV-cache distillation, superposition, and looped computation [23, 10, 63]. Concurrently, several methods extend these techniques to multimodal tasks, preserving visual cues within latent representations [59, 24, 44, 28, 20]. Despite their differences, these methods all treat continuous latent tokens as the medium for intermediate reasoning semantics.

2.2 Chain-of-Thought Compression and Visual Compression

As reasoning models produce increasingly long chains of thought, another line of work investigates how to compress these traces with minimal loss of performance. In the text domain, existing methods often employ preference optimization [62] or reward signals that encourage conciseness [47]. Inspired by the optical compression of DeepSeek-OCR [49], recent approaches use rendered-text compression to connect explicit CoT with latent computation. Specifically, Render-of-Thought [46] uses visual embeddings of rendered rationales for supervision; OneLatent [30] compresses these into a single token; and ReGuLaR [43] treats rendered CoT as a visual-semantic prior. These studies indicate that effective compression requires encoding reasoning steps into denser, and continuous representations.

2.3 Unified Model

Unified multimodal models represent heterogeneous modalities within a single generative architecture. Chameleon [4] models interleaved sequences via early fusion, while Emu3 [45] extends next-token prediction across modalities using a unified token space. Show-o and Janus [56, 54] further advance this concept through varied visual encoding strategies. Conceptually, these models align closely with vector-quantized latent modeling: VQ-VAE establishes how continuous representations can be discretized into codebook entries that retain semantic structure while remaining compatible with autoregressive symbolic modeling [41]. Building on this principle, rather than unifying external modalities, our method discretizes internal reasoning states into vocabulary-level tokens, enabling latent computation directly within standard autoregressive language modeling.

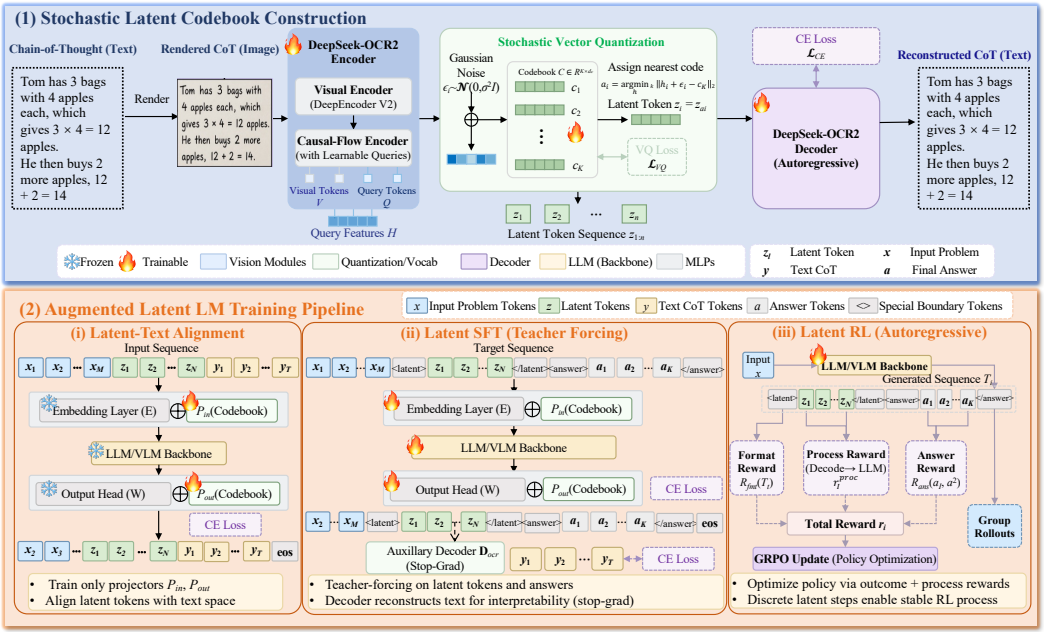


Figure 2: Overview of **DLR**. **Top**: we render chain-of-thought traces into images, encode them with a DeepSeek-OCR2-based visual compressor, and learn a stochastic vector-quantized latent codebook jointly with an OCR decoder to obtain semantically recoverable latent tokens. **Bottom**: the learned latent vocabulary is projected into the LLM token space and used to train an augmented latent language model through latent-text alignment, supervised fine-tuning, and reinforcement learning.

3 Methods

3.1 Overview

Discrete Latent Reasoning (DLR) bridges the optimization gap between continuous latent reasoning and standard discrete autoregressive LLM training. Instead of relying on continuous vectors, **DLR** represents intermediate reasoning states as explicit discrete latent tokens within a unified sequence. As illustrated in Fig. 2, our method features two main components: a **stochastic latent codebook construction** module and an **augmented latent LM training procedure**.

The first component constructs a stable latent vocabulary by extracting and discretizing features from rendered CoT traces. We encode these visual traces using a DeepSeek-OCR2 encoder [50], whose learnable queries extract causally ordered, structurally stable features ideal for vector quantization. These dense visual features are then mapped into a vector-quantized codebook. Joint training with the DeepSeek-OCR2 decoder ensures these discrete tokens retain high semantic fidelity and remain textually reconstructable. This process yields a structured latent vocabulary, denoted as $\mathcal{V}_{\text{lat}} = \{z_1, \ldots, z_K\}$, ready for LLM integration.

The second component trains the LLM to autonomously generate and reason over these latent tokens alongside natural language. We expand the text vocabulary $\mathcal{V}_{\text{text}}$ by using lightweight projectors to map codebook vectors into the LLM's dimensionality, yielding latent embedding and output matrices E_{lat} and W_{lat}. By merging the vocabularies ($\mathcal{V} = \mathcal{V}_{\text{text}} \cup \mathcal{V}_{\text{lat}}$) and concatenating their weights ($E = [E_{\text{text}}; E_{\text{lat}}], W = [W_{\text{text}}, W_{\text{lat}}]$), the model naturally performs standard next-token prediction $p(s_t \mid s_{<t}) = \text{softmax}(W^\top u_t)$ over the unified space. Training proceeds in three stages: ***(i) latent-text alignment*** to synchronize embeddings; ***(ii) latent SFT*** to teach latent trajectory generation; and ***(iii) latent RL*** to optimize the policy via process and outcome rewards.

3.2 Stochastic Latent Codebook Construction

The latent codebook converts rendered reasoning traces into reusable discrete tokens while preserving semantic recoverability throught feature extraction, vector quantization, and codebook optimization.

Causal-Flow Rendered Feature Extraction. Given a textual chain-of-thought $y = (y_1, \ldots, y_T)$, we first render it into an image $I = \mathcal{R}(y)$ and encode it using a DeepSeek-OCR2 encoder ($\mathbf{E}_{\text{vis}}$). The encoder maps I into compressed visual tokens $V = \mathbf{E}_{\text{vis}}(I) \in \mathbb{R}^{N_v \times d_c}$. To extract causally ordered semantics, we maintain a pool of position-aware learnable queries and dynamically truncate it to match the visual token length, yielding $Q \in \mathbb{R}^{N_v \times d_c}$. The concatenated sequence $[V; Q]$ is processed by a causal transformer $\mathbf{E}_{\text{cf}}$, and we select the query-side outputs as our continuous reasoning features $H = [h_1, \ldots, h_{N_v}] = \Pi_Q(\mathbf{E}_{\text{cf}}([V; Q]))$, where Π_Q retrieves the last N_v positions. This yields a causally ordered, visually compressed representation of the trace [50].

Stochastic Vector Quantization. We introduce a vector-quantized codebook $C = \{c_1, \ldots, c_K\}$ with $c_k \in \mathbb{R}^{d_c}$, which is warm-started via clustering on reasoning corpora to prevent early collapse. Each code vector c_k corresponds to a discrete latent token z_k in the vocabulary. To map continuous features h_i to discrete latent tokens with stability while improving codebook utilization, we employ stochastic quantization by injecting Gaussian perturbation $\epsilon_i \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ into h_i. The discrete index a_i is then assigned via minimum L_2 distance:

$$a_i = \arg\min_{k \in \{1,\ldots,K\}} \|h_i + \epsilon_i - c_k\|_2. \tag{1}$$

The continuous feature is replaced by the corresponding code vector to form the quantized feature $\hat{h}_i = c_{a_i}$, and is then mapped to the discrete latent token $z_i = z_{a_i}$. This process yields the quantized embedding sequence $\hat{H} = [\hat{h}_1, \ldots, \hat{h}_{N_v}]$ and the discrete token sequence $z_{1:N_v}$.

Curriculum-based Codebook Optimization. The codebook is trained to ensure that the discrete tokens retain interpretable semantics. To allow gradients to propagate through the discrete assignment, we use the straight-through estimator to form the differentiable quantized embeddings $\hat{h}_i^{\text{st}} = h_i + \text{sg}[\hat{h}_i - h_i]$, with the sequence denoted as $\hat{H}^{\text{st}}$, where $\text{sg}[\cdot]$ is the stop-gradient operator. The decoder of DeepSeek-OCR2 ($\mathbf{D}_{\text{ocr}}$) is trained to reconstruct the original text y from $\hat{H}^{\text{st}}$ using

$$\mathcal{L}_{\text{CE}}(\hat{H}^{\text{st}}, y) = -\sum_{t=1}^{T} \log \mathbf{D}_{\text{ocr}}(y_t \mid y_{<t}, \hat{H}^{\text{st}}). \tag{2}$$

The discrete codebook assignments are learned via a standard vector-quantization loss:

$$\mathcal{L}_{\text{VQ}} = \sum_{i=1}^{N_v} \|\text{sg}[h_i] - \hat{h}_i\|_2^2 + \beta \sum_{i=1}^{N_v} \|h_i - \text{sg}[\hat{h}_i]\|_2^2. \tag{3}$$

To stabilize early training, we employ a two-branch curriculum where the decoder receives both continuous features H and quantized features $\hat{H}^{\text{st}}$. The step-e training loss is:

$$\mathcal{L}_{\text{cb}}^{(e)} = \mathcal{L}_{\text{CE}}(\hat{H}^{\text{st}}, y) + \alpha(e)\mathcal{L}_{\text{CE}}(H, y) + \lambda_{\text{VQ}}\mathcal{L}_{\text{VQ}}. \tag{4}$$

The annealing factor $\alpha(e)$ decays from 1 to 0 over the first epoch. After convergence, the continuous branch is entirely discarded, forcing the OCR decoder to act purely as an interpretability module reading from the discrete latent trajectory.

3.3 Augmented Latent Language Model Training Pipeline

After constructing the latent codebook, the LM is trained over the augmented vocabulary $\mathcal{V}_{\text{text}} \cup \mathcal{V}_{\text{lat}}$, transitioning through three stages: latent-text alignment, latent SFT, and latent RL.

3.3.1 Latent-Text Alignment

The goal of this stage is to align the newly introduced latent tokens with the pretrained text manifold. We freeze the LLM backbone and update only the lightweight input/output projectors. Given a reasoning trace y, we quantize its rendered features into a latent sequence $z_{1:n}$. We then construct a concatenated sequence $s = [z_{1:n}, y_{1:T}]$ and optimize the projectors using a standard autoregressive cross-entropy loss over the entire sequence. This enforces the latent prefix to encode sufficient semantic context, enabling the frozen LLM to continue generating the corresponding explicit text.

3.3.2 Latent Supervised Fine-Tuning

During SFT, the model learns to autonomously generate latent reasoning trajectories conditioned on task inputs. To support structured generation and format verification during the subsequent RL stage, we introduce special boundary tokens. For a problem x, reasoning trace y, and answer a, we construct the target sequence using the trained codebook as:

$$s = [x; <\texttt{latent}>; z_{1:n}; </\texttt{latent}>; <\texttt{answer}>; a_{1:M}; </\texttt{answer}>]. \tag{5}$$

The LLM is optimized via standard next-token cross-entropy over the entire generated portion, encompassing latent tokens, final answer, and boundaries:

$$\mathcal{L}_{\text{SFT}} = -\sum_{t=|x|+1}^{|s|} \log p_\theta(s_t \mid s_{<t}). \tag{6}$$

To preserve the interpretability of latent trajectories without compromising training stability, we introduce a dual-branch parallel training. While the LLM undergoes teacher-forced training on $z_{1:n}$, the continuous input embeddings of the latent tokens, denoted as $G(z_{1:n}) = [E_{\text{lat}}(z_1), \ldots, E_{\text{lat}}(z_n)]$, are also fed into the auxiliary latent decoder $\mathbf{D}_{\text{ocr}}$, which is trained to reconstruct the text y:

$$\mathcal{L}_{\text{dec}} = -\sum_{t=1}^{T} \log p_{\mathbf{D}_{\text{ocr}}}\big(y_t \mid y_{<t}, \text{sg}[G(z_{1:n})]\big). \tag{7}$$

By applying a stop-gradient operator ($\text{sg}[\cdot]$) at the LLM interface, we structurally isolate the two processes. This ensures the LLM independently masters latent reasoning efficiency, while the auxiliary decoder translates these latent states back into interpretable steps without interfering with the LLM's representation space.

3.3.3 Latent Reinforcement Learning

In the final stage, we optimize the latent policy π_θ to refine accuracy and intermediate structural coherence. Unlike SFT, the full response trajectory $\tau_i \sim \pi_\theta(\cdot \mid x)$ is generated freely without teacher forcing. We parse τ_i based on the boundary tokens to extract the latent trajectory $z^{(i)}_{1:n_i}$ and the predicted answer $a^{(i)}_{1:M_i}$. We evaluate each rollout using a composite reward function.

To ensure the quality of the latent trajectories, we propose the **Process Alignment Reward**. Instead of only rewarding the final answer, we also leverage the auxiliary decoder to translate the latent trajectory into an interpretable trace $\hat{y}^{(i)} = \mathbf{D}_{\text{ocr}}\big(G(z^{(i)}_{1:n_i})\big)$. We then prompt a frozen evaluator LLM (π_{eval}, *e.g.*, the initial instruction-tuned checkpoint before latent-text alignment) with this decoded trace to predict a verification answer $\tilde{a}^{(i)}$, which is compared against the ground-truth a^*:

$$\tilde{a}^{(i)} \sim \pi_{\text{eval}}(\cdot \mid x, \hat{y}^{(i)}), \quad r_i^{\text{proc}} = R_{\text{ans}}(\tilde{a}^{(i)}, a^*). \tag{8}$$

This mechanism provides a dynamic, semantic-level verification of the latent reasoning process. The total reward for rollout i combines this process reward with the final answer correctness R_{ans} and a format verification score R_{fmt}. Specifically, R_{fmt} ensures the model strictly adheres to the structural constraints enforced during SFT by verifying the exact presence and ordering of the boundary tokens:

$$r_i = R_{\text{ans}}(a^{(i)}, a^*) + \lambda_{\text{proc}} r_i^{\text{proc}} + \lambda_{\text{fmt}} R_{\text{fmt}}(\tau_i). \tag{9}$$

With the composite reward defined, we optimize the policy using the Group Relative Policy Optimization (GRPO) algorithm [15]. Following standard GRPO practices, we apply group-based advantage normalization, clipped probability ratios to prevent destructive policy updates, and a KL-divergence penalty against the reference policy π_{ref}. This concise setup maximizes the expected reward while maintaining training stability, avoiding the need for a separate value model.

4 Experiments

4.1 Experimental Setup

Training Data. DLR has two separate data pipelines: one for latent codebook training on rendered CoT images, and another for latent LLM training on mixed text-latent sequences (details in App. A.1).

Table 1: Experimental results on four grade-school reasoning benchmarks with Qwen3-4B (LLM) and Qwen3-VL-4B-Instruct (LMM). * indicate the Pass@1 significantly outperform the second-best latent reasoning method ($p < 0.05$). The results of baseline methods are from CoLaR or RoT.

	GSM8K-Aug		GSM-Hard		SVAMP		MultiArith		Average	
Method	Acc.	#L	Acc.	#L	Acc.	#L	Acc.	#L	Acc.	#L
Qwen3-4B										
Direct-SFT	$27.4_{\pm.4}$	$0.0_{\pm.0}$	$9.2_{\pm.2}$	$0.0_{\pm.0}$	$69.4_{\pm.4}$	$0.0_{\pm.0}$	$86.3_{\pm.3}$	$0.0_{\pm.0}$	48.1	0.0
CoT-SFT	$83.1_{\pm.3}$	$119.8_{\pm.8}$	$52.3_{\pm.3}$	$189.4_{\pm1.1}$	$82.9_{\pm.3}$	$53.7_{\pm.7}$	$98.5_{\pm.3}$	$57.6_{\pm.8}$	79.2	105.1
iCoT	$13.5_{\pm.2}$	$0.0_{\pm.0}$	$4.1_{\pm.2}$	$0.0_{\pm.0}$	$36.9_{\pm.2}$	$0.0_{\pm.0}$	$49.2_{\pm.7}$	$0.0_{\pm.0}$	25.9	0.0
Coconut (CoLM 2025)	$16.9_{\pm.3}$	$6.0_{\pm.0}$	$5.4_{\pm.3}$	$6.0_{\pm.0}$	$43.6_{\pm.5}$	$6.0_{\pm.0}$	$60.3_{\pm.7}$	$6.0_{\pm.0}$	31.6	6.0
CODI (EMNLP 2025)	$7.3_{\pm.5}$	$6.0_{\pm.0}$	$2.2_{\pm.2}$	$6.0_{\pm.0}$	$11.0_{\pm.6}$	$6.0_{\pm.0}$	$18.3_{\pm.8}$	$6.0_{\pm.0}$	9.7	6.0
CoLaR (NeurIPS 2025)	$40.0_{\pm.2}$	$39.6_{\pm.1}$	$9.2_{\pm.1}$	$47.4_{\pm.2}$	$57.7_{\pm.2}$	$19.2_{\pm.1}$	$82.2_{\pm.1}$	$21.1_{\pm.1}$	47.3	31.8
DLR (SFT)	$\mathbf{47.4}_{\pm.4}$	$6.4_{\pm.3}$	$\mathbf{13.7}_{\pm.3}$	$6.8_{\pm.2}$	$\mathbf{70.3}_{\pm.2}$	$3.6_{\pm.3}$	$\mathbf{96.1}_{\pm.3}$	$4.2_{\pm.1}$	**56.9**	5.3
DLR (Full)	$\mathbf{62.2}^{*}_{\pm.4}$	$6.5_{\pm.2}$	$\mathbf{19.3}^{*}_{\pm.3}$	$7.0_{\pm.3}$	$\mathbf{72.3}^{*}_{\pm.2}$	$3.8_{\pm.2}$	$\mathbf{97.8}^{*}_{\pm.3}$	$4.3_{\pm.3}$	**62.9**	5.4
Qwen3-VL-4B-Instruct										
Direct-SFT	$26.2_{\pm.3}$	$0.0_{\pm.0}$	$9.5_{\pm.1}$	$0.0_{\pm.0}$	$70.0_{\pm.3}$	$0.0_{\pm.0}$	$85.6_{\pm.4}$	$0.0_{\pm.0}$	47.8	0.0
CoT-SFT	$81.2_{\pm.4}$	$127.3_{\pm1.0}$	$53.4_{\pm.3}$	$191.1_{\pm1.6}$	$84.3_{\pm.3}$	$55.9_{\pm.6}$	$98.3_{\pm.4}$	$59.1_{\pm.6}$	79.3	108.4
Coconut (CoLM 2025)	$16.1_{\pm.4}$	$6.0_{\pm.0}$	$5.2_{\pm.3}$	$6.0_{\pm.0}$	$42.0_{\pm.3}$	$6.0_{\pm.0}$	$58.9_{\pm.2}$	$6.0_{\pm.0}$	30.5	6.0
CODI (EMNLP 2025)	$7.1_{\pm.3}$	$6.0_{\pm.0}$	$2.0_{\pm.4}$	$6.0_{\pm.0}$	$10.7_{\pm.1}$	$6.0_{\pm.0}$	$17.8_{\pm.2}$	$6.0_{\pm.0}$	9.4	6.0
CoLaR (NeurIPS 2025)	$39.2_{\pm.2}$	$38.8_{\pm.5}$	$8.2_{\pm.4}$	$47.8_{\pm.7}$	$52.0_{\pm.4}$	$18.4_{\pm.6}$	$81.0_{\pm.3}$	$22.1_{\pm.3}$	45.1	31.8
RoT (ACL 2026)	$37.8_{\pm.3}$	$32.0_{\pm.0}$	$14.1_{\pm.2}$	$32.0_{\pm.0}$	$72.7_{\pm.4}$	$32.0_{\pm.0}$	$97.2_{\pm.4}$	$32.0_{\pm.0}$	55.4	32.0
DLR (SFT)	$\mathbf{48.0}_{\pm.4}$	$6.3_{\pm.4}$	$\mathbf{14.7}_{\pm.3}$	$6.8_{\pm.5}$	$\mathbf{68.7}_{\pm.2}$	$3.6_{\pm.1}$	$\mathbf{97.8}_{\pm.3}$	$4.2_{\pm.2}$	**57.3**	5.2
DLR (Full)	$\mathbf{63.3}^{*}_{\pm.3}$	$6.5_{\pm.1}$	$\mathbf{18.3}^{*}_{\pm.4}$	$7.0_{\pm.6}$	$\mathbf{72.7}_{\pm.3}$	$3.8_{\pm.4}$	$\mathbf{98.3}^{*}_{\pm.2}$	$4.3_{\pm.3}$	**63.2**	5.4

For **latent codebook training**, we aggregate $\sim$ 1M reasoning traces from GSM8K-Aug [11], MATH [18], and MathX-5M [31]. Textual CoTs (up to $2,048$ tokens) are rendered into adaptive square RGB images (up to 1024×1024 resolution), producing a maximum of 256 visual tokens. This achieves an $\sim 8 - 20\times$ compression ratio, efficiently shifting the computational reasoning overhead into the dense latent space. For **latent LM training**, we construct two configurations. The *base* setup utilizes 385K GSM8K-Aug [11] samples, allocating all 385K for latent-text alignment, 300K for SFT, and the remaining 85K for RL. To evaluate generalization, a *scaled* setup distributes $\sim$12K MATH [18] samples across the same three training stages.

Evaluation Data. For the *base* configuration, we evaluate zero-shot performance on four standard arithmetic benchmarks: GSM8K (1,319) [7], GSM-Hard (1,319) [13], SVAMP (1,000) [33], and MultiArith (600) [37]. The *scaled* configuration is evaluated on the harder MATH-500 benchmark (500) [18]. All evaluations are strictly designed to omit explicit chains-of-thought in the zero-shot setting (further details are provided in App. A.2).

Backbone Models. To assess both the scalability and architectural generalizability of our framework, we instantiate DLR across diverse backbones ranging from 1B to 8B parameters. For LLMs, we adopt Qwen3-4B [58], LLaMA-3.2-1B/3B [1], and LLaMA-3.1-8B [14]. For LMMs, we use Qwen3-VL-Instruct [2] at 2B, 4B, and 8B scales (details in App. B.1).

Baseline Methods. We benchmark **DLR** against representative latent reasoning methods. For LLMs, we compare against iCoT [12] (progressive CoT removal), Coconut [17] (continuous feedback), CODI [38] (self-distillation), CoLaR [40] (dynamic compression), and ReGuLaR [43] (variational rendering). For LMMs, we adapt Coconut, CODI, and CoLaR to accept visual inputs and include RoT [46], which aligns continuous states with rendered CoT embeddings. We report the mean $\pm$ standard deviation over five random seeds for Pass@1 (Acc.) and computational efficiency (#L, average token length of the reasoning chain) (details in App. B.2).

Implementation Details. All experiments are conducted on an $8\times$ NVIDIA H100 node in bfloat16 mixed precision. We train the DeepSeek-OCR2 latent codebook via Unsloth [16] (3 epochs, lr=$1e{-4}$) and optimize the language models via TRL [42] through a three-stage process: (1) *latent-text alignment* updates only the projectors for 3 epochs (lr=$1e{-4}$); (2) *latent SFT* updates the backbone, projectors, and codebook for 3 epochs (lr=$2e{-5}$), training the auxiliary decoder in parallel via stop-gradient; (3) *latent RL* updates the backbone and projectors using GRPO for 1 epoch (lr=$1e{-6}$, 32 rollouts/prompt). Further training configurations are detailed in App. C.

Table 2: Experimental results on MATH-500 with Qwen3-VL at three scales. We compare **DLR** against SFT-w/o CoT, explicit SFT-CoT, and the rendered-CoT baseline RoT.

	Qwen3-VL-2B		**Qwen3-VL-4B**		**Qwen3-VL-8B**	
Method	Acc.	#L	Acc.	#L	Acc.	#L
Direct-SFT	$20.8_{\pm.2}$	$0.0_{\pm.0}$	$29.4_{\pm.3}$	$0.0_{\pm.0}$	$33.5_{\pm.5}$	$0.0_{\pm.0}$
CoT-SFT	$29.2_{\pm.3}$	$324.5_{\pm2.6}$	$55.8_{\pm.4}$	$291.5_{\pm1.9}$	$69.4_{\pm.3}$	$310.6_{\pm1.7}$
RoT	$24.0_{\pm.2}$	$64.0_{\pm.0}$	$33.2_{\pm.4}$	$64.0_{\pm.0}$	$37.8_{\pm.3}$	$64.0_{\pm.0}$
DLR (SFT)	$\mathbf{29.4}_{\pm.3}$	$26.7_{\pm.8}$	$\mathbf{35.4}_{\pm.3}$	$27.9_{\pm.7}$	$\mathbf{39.8}_{\pm.4}$	$27.2_{\pm.5}$
DLR (Full)	$\mathbf{34.0}^{*}{}_{\pm.3}$	$28.9_{\pm.6}$	$\mathbf{44.6}^{*}{}_{\pm.4}$	$28.4_{\pm.5}$	$\mathbf{54.0}^{*}{}_{\pm.3}$	$29.5_{\pm.6}$

Table 3: Ablation on training-stage composition with Qwen3-VL-4B-Instruct. ✓indicate which components are enabled: Stage 1 (latent-text alignment), Stage 2 (latent SFT), Stage 3 (latent RL), and the process alignment reward, r_i^{proc} (Proc.) inside Stage 3. The last row is our full configuration.

	Components				**GSM8K-Aug**		**GSM-Hard**		**SVAMP**		**MultiArith**	
Method	**S1**	**S2**	**S3**	**Proc.**	Acc.	#L	Acc.	#L	Acc.	#L	Acc.	#L
– Align	✗	✓	✓	✓	$49.6_{\pm.2}$	6.4	$14.3_{\pm.3}$	6.6	$65.3_{\pm.2}$	3.7	$93.9_{\pm.3}$	4.1
– SFT	✓	✗	✓	✓	$21.4_{\pm.3}$	8.5	$3.2_{\pm.5}$	11.3	$42.3_{\pm.4}$	9.7	$49.4_{\pm.6}$	8.4
– RL	✓	✓	✗	✓	$48.0_{\pm.4}$	6.3	$14.7_{\pm.3}$	6.8	$68.7_{\pm.2}$	3.6	$97.8_{\pm.3}$	4.2
– Proc. Rwd.	✓	✓	✓	✗	$59.6_{\pm.2}$	6.6	$16.2_{\pm.3}$	6.7	$72.3_{\pm.1}$	3.6	$98.3_{\pm.4}$	4.3
DLR (Full)	✓	✓	✓	✓	$\mathbf{63.3}_{\pm.3}$	6.5	$\mathbf{18.3}_{\pm.4}$	7.0	$\mathbf{72.7}_{\pm.3}$	3.8	$\mathbf{98.3}_{\pm.2}$	4.3

4.2 Main Results

We evaluate **DLR** along two complementary axes. First, we benchmark performance and out-of-domain (OOD) generalization on grade-school reasoning tasks (GSM8K-Aug, GSM-Hard, SVAMP, MultiArith) using 4B LLMs and LMMs (Tab. 1). Second, we assess its scalability and reasoning capability on the high-school-level MATH-500 benchmark across three model scales (Tab. 2). **Additional experiments on different LLM and LMM backbones are detailed in App. D.**

Results on Grade-School-Level Reasoning and OOD Generalization. Tab. 1 compares **DLR** against baselines on both LLM and LMM backbones. On the in-domain GSM8K-Aug dataset, **DLR** achieves the highest accuracy among latent methods while **reducing the reasoning length** to $\sim$6 tokens (versus 119 in CoT-SFT and 32 in RoT). On OOD benchmarks like GSM-Hard and SVAMP, continuous latent baselines suffer from severe error accumulation and representation drift. In contrast, **DLR** demonstrates **strong out-of-domain generalization**, significantly outperforming the second-best baseline ($p < 0.05$) while maintaining a superior compression rate. This confirms that our discrete codebook provides stable symbolic anchors that prevent latent drift, regardless of whether the underlying backbone is an LLM or LMM.

Results on High-School-Level Scalability. To examine scalability on harder reasoning tasks, Tab. 2 reports results on MATH-500 across three LMM scales (2B, 4B, 8B). As capacity increases, **DLR** exhibits **consistent scaling behavior**. While the rendered-CoT baseline (RoT) shows diminishing returns from 4B to 8B (+4.6 points), **DLR (Full)** consistently scales, achieving a +9.4 point gain to reach 54.0% accuracy on the 8B backbone. Comparing **DLR (SFT)** to **DLR (Full)** highlights a **performance boost from latent RL**, confirming that discrete tokens reliably support reward-driven policy optimization. Moreover, **DLR** achieves this using fewer than 30 latent tokens, offering much higher compression than RoT (64) and explicit CoT ($\sim$300), demonstrating that our discrete vocabulary efficiently unlocks semantic capacity without acting as an information bottleneck.

4.3 Ablation Study

All ablations are conducted on Qwen3-VL-4B-Instruct and evaluated on the four grade-school benchmarks. **Further ablations on the latent codebook and latent decoding are in App. E.**

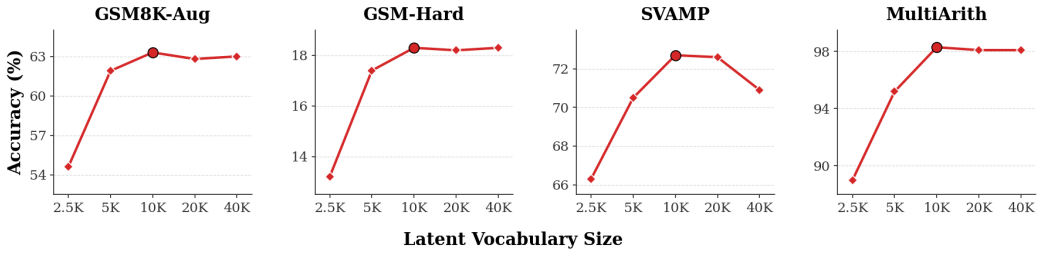


Figure 3: Effect of latent vocabulary size (K) on reasoning performance.

Latent Vocabulary Capacity. We first study the effect of the latent codebook size by varying $K \in \{2.5\text{K}, 5\text{K}, 10\text{K}, 20\text{K}, 40\text{K}\}$ (Fig. 3). Performance follows a clear two-phase trend: from $K = 2.5\text{K}$ to 10K, accuracy steadily improves across all four benchmarks as the vocabulary acquires sufficient capacity to capture diverse reasoning semantics. However, from $K = 10\text{K}$ to 40K, performance plateaus and occasionally degrades slightly, suggesting that an excessively large codebook leads to under-utilized codes and limits generalization. Thus, we set $K = 10\text{K}$ to balance representation capacity and training stability.

Training-Stage Composition. Tab. 3 analyzes the contribution of our three-stage process and the specific reward design. Removing Stage-1 Alignment ($-$ Align) forces the latent tokens to co-train with the backbone from scratch, failing to ground them semantically and causing a major performance drop. Skipping Stage-2 SFT ($-$ SFT) causes a performance collapse, indicating that RL cannot learn complex latent distributions from sparse outcome signals alone. Furthermore, while the base Stage-3 RL provides clear refinement over SFT ($-$ RL), disabling the *Process Alignment Reward* ($-$ Proc. Rwd.) strictly degrades accuracy. This demonstrates that the process reward is critical for keeping the latent trajectory semantically structured during policy optimization.

4.4 Interpretability through Case Study

Fig. 4 illustrates how our auxiliary decoder translates latent trajectories into interpretable steps. The decoded chains of thought capture the correct mathematical logic and consistently yield accurate final answers. While the decoder occasionally hallucinates trivial semantic details during reconstruction, such as substituting incorrect names (*e.g.*, "Ethan" for "Eliza") or units (*e.g.*, "miles" for "meters"), these minor artifacts are inconsequential and do not disrupt the underlying causal reasoning of the latent states. This validates the strong interpretability of our discrete latent method. Additional case studies are provided in App. F.

5 Conclusion

In this paper, we resolve the misalignment between continuous latent states and discrete autoregressive supervision by proposing **Discrete Latent Reasoning (DLR)**. **DLR** represents intermediate reasoning steps as explicit discrete tokens using a visual-semantic codebook constructed from rendered chains of thought. To integrate these tokens, we design a three-stage training process with dual-branch supervised fine-tuning and a novel Process Alignment Reward, maintaining both training stability and trajectory interpretability. Extensive experiments across diverse LLM and LMM architectures confirm that **DLR** achieves superior accuracy, strong out-of-domain generalization, and consistent scaling on mathematical benchmarks. By dynamically compressing the reasoning length while retaining semantic interpretability, **DLR** offers an efficient, controllable, and scalable approach that advances the development of latent reasoning models.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 62471287).

GSM8K-Aug-Case1

Question: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?

Decoded CoT: He runs 3 sprints for 3 days a week, so he runs 3*60 = 180 miles in a sprint. In 3 times, he runs 180*3 = 540 miles.

GT CoT: He sprints 3*3=9 times. So he runs 9*60=540 meters.

Answer: The answer is: 540.

GSM8K-Aug-Case2

Question: Eliza's rate per hour for the first 40 hours she works each week is $10. She also receives an overtime pay of 1.2 times her regular hourly rate. If Eliza worked for 45 hours this week, how much are her earnings for this week?

Decoded CoT: Ethan has 40*10 = 400. Together, they have 400+extra comic pays. They have 5 hours each week, so they have 5*12 = 60. They will have 400+60 left.

GT CoT: Eliza is entitled to 45 -40 = 5 hours overtime pay. Her hourly rate for the overtime pay is $10 x 1.2 = $12. So, Eliza will receive $12 x 5 =$60 for overtime pay. Her regular weekly earning is $10 x 40 = $400. Thus, Eliza will receive a total of $400 + $60 = $460 for this week's work.

Answer: The answer is: 460.

Figure 4: Case study of latent-to-text decoding. Best viewed by zoom in.

References

[1] AI@Meta. Llama 3.2 model card. `https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_2/`, 2024. Accessed: 2026-05-06.

[2] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020.

[4] Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024. doi: 10.48550/arXiv.2405.09818.

[5] Shuochen Chang, Tong Bai, Xiaofeng Zhang, Qianli Ma, Qingyang Liu, Zhaohe Liao, Yibo Miao, and Li Niu. Unlocking the black box of latent reasoning: An interpretability-guided approach to intervention. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL 2026)*, 2026.

[6] Yuzhe Chen, Jiale Cao, Xuyang Liu, Jin Xie, Aiping Yang, and Yanwei Pang. Stdec: Spatio-temporal stability guided decoding for dllms. *arXiv preprint arXiv:2604.06330*, 2026.

[7] 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.

[8] Yingqian Cui, Zhenwei Dai, Bing He, Zhan Shi, Hui Liu, Rui Sun, Zhiji Liu, Yue Xing, Jiliang Tang, and Benoit Dumoulin. How do latent reasoning methods perform under weak and strong supervision? *arXiv preprint arXiv:2602.22441*, 2026.

[9] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024.

[10] 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.

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

[12] Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *URL https://arxiv.org/abs/2405.14838*, 2024.

[13] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR, 2023.

[14] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[15] Daya Guo et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z.

[16] Daniel Han, Michael Han, and Unsloth team. Unsloth. `https://github.com/unslothai/unsloth`, 2023. Software library.

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

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

[19] Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8003–8017, 2023. doi: 10.18653/v1/2023.findings-acl.507.

[20] Byungwoo Jeon, Yoonwoo Jeong, Hyunseok Lee, Minsu Cho, and Jinwoo Shin. Vision-aligned latent reasoning for multi-modal large language model. *arXiv preprint arXiv:2602.04476*, 2026. doi: 10.48550/arXiv.2602.04476.

[21] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[22] Tomek Korbak, Mikita Balesni, Elizabeth Barnes, Yoshua Bengio, Joe Benton, Joseph Bloom, Mark Chen, Alan Cooney, Allan Dafoe, Anca Dragan, et al. Chain of thought monitorability: A new and fragile opportunity for ai safety. *arXiv preprint arXiv:2507.11473*, 2025.

[23] Anna Kuzina, Maciej Pióro, and Babak Ehteshami Bejnordi. Kava: Latent reasoning via compressed kv-cache distillation. In *International Conference on Learning Representations*, 2026.

[24] Bangzheng Li, Ximeng Sun, Jiang Liu, Ze Wang, Jialian Wu, Xiaodong Yu, Emad Barsoum, Muhao Chen, and Zicheng Liu. Latent visual reasoning. In *International Conference on Learning Representations*, 2026.

[25] Xiaoyuan Li, Yuzhe Wang, Moxin Li, Keqin Bao, Rui Men, Yichang Zhang, Dayiheng Liu, Wenjie Wang, and Fuli Feng. Sage: Scalable automated robustness augmentation for llm knowledge evaluation. *arXiv preprint arXiv:2605.12022*, 2026.

[26] You Li, Chi Chen, Yanghao Li, Fanhu Zeng, Kaiyu Huang, Jinan Xu, and Maosong Sun. Imagination helps visual reasoning, but not yet in latent space. *arXiv preprint arXiv:2602.22766*, 2026. doi: 10.48550/arXiv.2602.22766.

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

[28] Chengzhi Liu, Yuzhe Yang, Yue Fan, Qingyue Wei, Sheng Liu, and Xin Eric Wang. Reasoning within the mind: Dynamic multimodal interleaving in latent space. *arXiv preprint arXiv:2512.12623*, 2025. doi: 10.48550/arXiv.2512.12623.

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[30] Bo Lv, Yasheng Sun, Junjie Wang, and Haoxiang Shi. Onelatent: Single-token compression for visual latent reasoning. *arXiv preprint arXiv:2602.13738*, 2026. doi: 10.48550/arXiv.2602.13738.

[31] Modotte. MathX-5M. `https://huggingface.co/datasets/Modotte/MathX-5M`, 2026. Hugging Face dataset.

[32] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744, 2022.

[33] 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*, pages 2080–2094. Association for Computational Linguistics, 2021.

[34] Xiaohan Qin, Victor Wang, Ning Liao, Cancheng Zhang, Xiangdong Zhang, Mingquan Feng, Jingzhi Wang, and Junchi Yan. sstoken: Self-modulated and semantic-aware token selection for llm fine-tuning. In *International Conference on Learning Representations*, volume 2026, pages 79347–79368, 2026.

[35] Xiaohan Qin, Xiangdong Zhang, Tuo Dai, Sunan Zou, Huaijin Wu, Yebin Yang, Zhuo Xia, Lin Yao, Yuliang Liu, and Junchi Yan. Multi-stream attention residuals. *arXiv preprint arXiv:2610.13630*, 2026.

[36] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020.

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

[38] 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*, pages 677–693, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.emnlp-main.36.

[39] StepFun Team. Inside Step 5 Preview: Architecture and Training for Agentic Work. `https://platform.stepfun.ai/blog/inside-step-5-preview`, October 2026. StepFun technical blog, published October 15, 2026.

[40] Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of llm reasoning chains. In *Advances in Neural Information Processing Systems*, 2025.

[41] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, 2017.

[42] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformers reinforcement learning. `https://github.com/huggingface/trl`, 2020. Software library.

[43] Fanmeng Wang, Haotian Liu, Guojiang Zhao, Hongteng Xu, and Zhifeng Gao. ReGuLaR: Variational latent reasoning guided by rendered chain-of-thought. *arXiv preprint arXiv:2601.23184*, 2026.

[44] Qixun Wang, Yang Shi, Yifei Wang, Yuanxing Zhang, Pengfei Wan, Kun Gai, Xianghua Ying, and Yisen Wang. Monet: Reasoning in latent visual space beyond images and language. *arXiv preprint arXiv:2511.21395*, 2025. doi: 10.48550/arXiv.2511.21395.

[45] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.

[46] Yifan Wang, Shiyu Li, Peiming Li, Xiaochen Yang, Yang Tang, and Zheng Wei. Render-of-thought: Rendering textual chain-of-thought as images for visual latent reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL 2026)*, 2026.

[47] Yuhao Wang, Xiaopeng Li, Cheng Gong, Ziru Liu, Suiyun Zhang, Rui Liu, and Xiangyu Zhao. Efficient reasoning via reward model. *arXiv preprint arXiv:2511.09158*, 2025. doi: 10.48550/arXiv.2511.09158.

[48] Zhuohan Wang, Haoran Ma, Tianyu Wu, Yuanlin Duan, Zichun Liao, and Jieming Yu. Solving every step is not enough: Milestone oracles reveal a composition gap in llm math reasoning. *arXiv preprint arXiv:2609.32235*, 2026.

[49] Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr: Contexts optical compression. *arXiv preprint arXiv:2510.18234*, 2025. doi: 10.48550/arXiv.2510.18234.

[50] Haoran Wei, Yaofeng Sun, and Yukun Li. Deepseek-ocr 2: Visual causal flow. *arXiv preprint arXiv:2601.20552*, 2026.

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

[52] 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*, 2026.

[53] Jonathan Williams and Esin Tureci. Prioritize the process, not just the outcome: Rewarding latent thought trajectories improves reasoning in looped language models. *arXiv preprint arXiv:2602.10520*, 2026.

[54] Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, and Ping Luo. Janus: Decoupling visual encoding for unified multimodal understanding and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

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

[56] Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. In *International Conference on Learning Representations*, 2025.

[57] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. corr, abs/2407.10671, 2024. doi: 10.48550. *arXiv preprint ARXIV.2407.10671*, 2024.

[58] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[59] Zeyuan Yang, Xueyang Yu, Delin Chen, Maohao Shen, and Chuang Gan. Machine mental imagery: Empower multimodal reasoning with latent visual tokens. *arXiv preprint arXiv:2506.17218*, 2025. doi: 10.48550/arXiv.2506.17218.

[60] Xiangdong Zhang, Xiaohan Qin, Sunan Zou, Tuo Dai, Xiaoming Shi, Huaijin Wu, Yebin Yang, Zhuo Xia, Shaofeng Zhang, Lin Yao, et al. xhc: Expanded hyper-connections. *arXiv preprint arXiv:2607.14530*, 2026.

[61] Xiangdong Zhang, Debing Zhang, Shaofeng Zhang, Xiaohan Qin, Yu Cheng, and Junchi Yan. Nitp: Next implicit token prediction for llm pre-training. *arXiv preprint arXiv:2605.24956*, 2026.

[62] Yuxiang Zhang, Zhengxu Yu, Weihang Pan, Zhongming Jin, Qiang Fu, Deng Cai, Binbin Lin, and Jieping Ye. Tokensqueeze: Performance-preserving compression for reasoning llms. In *Advances in Neural Information Processing Systems*, 2025.

[63] Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025. doi: 10.48550/arXiv.2510.25741.

[64] Jiaxuan Zou, Yaozhong Xiong, and Yong Liu. Capabilities and fundamental limits of latent chain-of-thought. *arXiv preprint arXiv:2602.01148*, 2026.

This appendix provides additional materials to supplement the main submission. Section A details the data curation procedures, including the chain-of-thought rendering process and the specific formatting protocols for both codebook and language model training. Section B provides further details on the backbone architectures (both LLMs and LMMs) and elaborates on the mechanisms of the baseline latent reasoning methods used in our evaluation. Section C outlines the implementation details, including hardware setups, software frameworks, and hyperparameters for latent codebook construction and all three training stages of the latent language model. Section D presents additional comparative results on LLaMA-3.2-1B and QwenVL3-2B-Instruct to verify the effectiveness of the DLR method. Section E reports further ablation studies that analyze the impact of various hyperparameter choices, codebook sizes, and reward configurations on model performance. Section F provides extended interpretability analyses, showcasing qualitative examples of latent trajectories translated into explicit reasoning steps by our auxiliary OCR decoder. Section G discusses the current limitations of our approach and explores the broader societal impacts of deploying efficient, highly capable reasoning models.

A Details of Dataset

A.1 Details of Training Data

As outlined in the main text, **DLR** involves two distinct training stages that require separate data curation procedures: (i) the latent codebook, which is trained on rendered CoT images, and (ii) the latent language model, which is trained over mixed sequences of text and latent tokens. Below, we detail the data processing and formatting protocols for each stage.

A.1.1 Latent Codebook Training Data

For codebook training, we aggregate a corpus of approximately 1M reasoning traces covering a broad spectrum of mathematical difficulty. Specifically, we include the full GSM8K-Aug training split (385K) [11], the MATH training split (12K) [18] obtained by excluding MATH-500 from the full MATH dataset, and a stratified subsample of 603K instances from MathX-5M [31]. We extract the chain-of-thought segment from each instance and discard natural-language preambles. To ensure that every trace fits into a single rendered canvas, we retain only samples whose CoT length is at most $2,048$ text tokens, measured with the DeepSeek-OCR2 tokenizer [50]. Each retained CoT is rendered into a square RGB image via a Python pipeline, with the font size Gaussian-sampled in $[15, 20]$ pt and the canvas side length adaptively chosen to fit the rendered content. The resulting side length is clipped to the range $[64, 1024]$ pixels and resized to the nearest endpoint when it falls outside this range. Under the DeepEncoder V2 [50] tokenization scheme, the number of visual tokens produced from a side length L follows $N_{\text{vis}} = \left(\frac{L}{64}\right)^2$, so that a 1024×1024 canvas yields exactly 256 visual tokens. In the upper-bound setting, this corresponds to a $\sim 8\times$ compression ratio relative to the 2048-token CoT budget, consistent with the render-compression rationale of our approach. The codebook is then trained with the rendered image as input and the original textual CoT as decoding supervision for the auxiliary decoder $\mathbf{D}_{\text{ocr}}$.

A.1.2 Latent Language Model Training Data

For latent language model training, we construct two configurations. The *base* configuration uses only the GSM8K-Aug split [12], reusing the 385K rendered images and their codebook-derived latent token sequences. Within this split, the full set is used for Stage 1 latent-text alignment, the first 300K instances are used for Stage 2 supervised fine-tuning (SFT), and the remaining 85K instances are used for Stage 3 reinforcement learning (RL). During SFT, each sample is formatted as the triple $[x;\ z_{1:n};\ a_{1:M}]$, where $z_{1:n}$ is obtained from the pretrained codebook; during RL, only the problem x is fed to the model and the latent trajectory is generated autoregressively. The *scaled* configuration, designed to assess generalization to harder mathematical reasoning, uses the MATH training split [18] of ~ 12K instances, allocating the full set to alignment, 7.5K to SFT, and 4.5K to RL following the same protocol.

A.2 Details of Evaluation Data

To evaluate the arithmetic, out-of-domain, and advanced mathematical reasoning capabilities of our models, we conduct zero-shot evaluations across five standard benchmarks. No ground-truth explicit chains of thought are provided at test time. The detailed statistics and task descriptions for each evaluation dataset are as follows:

- **GSM8K** [7]: Consists of $1{,}319$ test problems, forming a widely used benchmark of grade-school math word problems that require multi-step arithmetic reasoning.
- **GSM-Hard** [13]: Contains $1{,}319$ test problems constructed by replacing numerical values in the original GSM8K test set with larger numbers, serving as a reliable metric to test whether the model relies on superficial heuristics or mathematical logic.
- **SVAMP** [33]: Comprises $1{,}000$ test problems, serving as a challenge set created by applying structural and linguistic variations to existing word problems to evaluate a model's stability against varying descriptions.
- **MultiArith** [37]: Includes 600 multi-step arithmetic problems. We use the full dataset as a standard out-of-domain (OOD) evaluation benchmark to measure the generalization capabilities of models trained primarily on GSM8K.
- **MATH-500** [18, 27]: Consists of 500 highly challenging test problems, forming a curated subset of the MATH dataset that features competition-level mathematics problems across advanced subjects such as algebra, geometry, and number theory.

B Details of Comparison Baselines

B.1 Details of Backbone LLMs and LMMs

As outlined in the main text, we instantiate our method across multiple backbone architectures to evaluate its generalizability and scalability. Specifically, we adopt the following model series:

- **LLaMA Series (3.2-1B/3B and 3.1-8B)** [1, 14]: As a representative open-weight LLM family developed by Meta, LLaMA offers highly optimized language models. We utilize LLaMA-3.2-1B/3B and LLaMA-3.1-8B to assess how discrete latent reasoning scales with model capacity within pure text domains.
- **Qwen3 Dense Text Models (4B)** [58]: Developed by Alibaba Cloud, the Qwen3 series provides state-of-the-art dense language models. We select the Qwen3-4B model as an alternative mid-scale text-only backbone, allowing us to evaluate the adaptability of our discrete latent method across different pre-training recipes and tokenizer designs.
- **Qwen3-VL-Instruct Series (2B, 4B, 8B)** [2]: To verify that our approach is agnostic to modality, we extend our evaluation to the large multimodal model (LMM) domain using the Qwen3-VL-Instruct family. Evaluating these native vision-language models at 2B, 4B, and 8B scales enables us to analyze the scalability of latent reasoning when processing visual representations and rendered reasoning traces.

B.2 Details of Compared Latent Reasoning Methods

To evaluate the effectiveness of our proposed method, we compare **DLR** against a representative set of state-of-the-art latent reasoning baselines, grouped by the modality of their underlying backbone:

Large Language Model Baselines:

- **iCoT** [12]: This method starts from a model trained on explicit chain-of-thought and progressively removes intermediate reasoning tokens during fine-tuning. This curriculum forces the model to internalize the reasoning process directly within its hidden states.
- **Coconut** [17]: A pioneering approach that operates in a purely continuous latent space. It feeds the last-layer hidden state back as the next input embedding, bypassing the LM head and tokenization under a multi-stage curriculum that gradually replaces textual steps with continuous thoughts.

Table 4: Detailed per-stage training configurations of **DLR**. "Trainable Modules" lists the parameters updated in each stage, while the rest are frozen. The symbol "$-$" indicates not applicable.

Hyperparameter	**Codebook**	**Stage 1: Align**	**Stage 2: SFT**	**Stage 3: RL (GRPO)**
Software Framework	Unsloth	TRL	TRL	TRL
DeepSpeed Stage	-	Zero-2	Zero-2	Zero-2
Optimizer	AdamW	AdamW	AdamW	AdamW
Peak Learning Rate	$1e{-4}$	$1e{-4}$	$2e{-5}$	$1e{-6}$
LR Schedule	Cosine Decay	Cosine Decay	Cosine Decay	Cosine Decay
Warmup Ratio	0.03	0.03	0.03	0.03
Weight Decay	$1e{-2}$	$1e{-2}$	$1e{-2}$	$1e{-2}$
Epochs	3	3	3	1
Max Sequence Length	256	256	256	256
Per-GPU Batch Size	8	8	8	8
Global Batch Size	64	64	64	64
Rollout Size (G)	–	–	–	32
PPO Inner Epochs	–	–	–	4
Sampling Temperature	–	–	–	1.0
KL Penalty (β_{KL})	–	–	–	$1e{-3}$
Reward Weights	–	–	–	$\lambda_{\mathrm{proc}} = 0.25, \lambda_{\mathrm{fmt}} = 0.1$
Trainable Modules	Enc. + C + $\mathbf{D}_{\mathrm{ocr}}$	$P_{\mathrm{in}}, P_{\mathrm{out}}$	LM + C + P_*(+ $\mathbf{D}_{\mathrm{ocr}}$)	LM + P_*
FLOPS	780P	208P	325P	1365P

- **CODI** [38]: This method casts implicit reasoning as a self-distillation problem. A single shared model is jointly trained under explicit and implicit CoT objectives, aligning the hidden activation at the answer token between the two modes so that continuous latent states inherit the semantics of explicit rationales.
- **CoLaR** [40]: This method compresses groups of explicit reasoning-token embeddings into dense latent tokens with a prompt-controllable compression factor, allowing the model to dynamically adjust its reasoning "speed" and computational cost at inference time.
- **ReGuLaR** [43]: ReGuLaR situates latent reasoning within a variational auto-encoding architecture. Each latent state is sampled from a posterior, and its prior is regularized by dense visual-semantic features obtained by rendering the corresponding CoT segment and encoding it with a frozen DeepSeek-OCR encoder.

Large Multimodal Model Baselines:

- **Render-of-Thought (RoT)** [46]: As the most directly comparable visual-rendering baseline to our method, RoT renders each CoT step into a single-line image. It then trains the model to produce continuous hidden states that align with the visual embeddings extracted by a frozen VLM encoder, effectively using rendered text as an external semantic anchor to supervise latent reasoning.

C Detailed Implementation Configurations

This section provides specifics regarding the hardware setups, software libraries, and hyperparameters used for both the latent codebook training and the three-stage latent language model training. A complete summary of the per-stage training configurations is provided in Table 4.

C.1 Frameworks and Hardware

All experiments are conducted on an $8\times$ NVIDIA H100 (80GB) node. To maximize throughput and memory efficiency, we employ mixed-precision `bfloat16` training across all stages. For the latent codebook, we leverage the Unsloth library [16] to optimize the DeepSeek-OCR2 architecture [50]. Its DeepEncoder V2 consists of a SAM-based [21] local encoder, a $16\times$ convolutional compressor, and a Qwen2 [57] causal-flow encoder, while the decoder $\mathbf{D}_{\mathrm{ocr}}$ follows the DeepSeek-LLM backbone. For the latent language model training, we utilize the TRL (Transformer Reinforcement Learning) library [42] integrated with DeepSpeed Zero-2 [36] to distribute optimizer states and gradients.

Furthermore, FlashAttention-2 [9] is enabled by default to accelerate self-attention computation for long multimodal sequences.

C.2 Latent Codebook Training Configurations

In accordance with the formulations in the main text, the entire codebook stack, including the visual encoder, the causal-flow module, the discrete codebook C, and the decoder $\mathbf{D}_{\mathrm{ocr}}$, is trained end-to-end. The latent codebook has a vocabulary size of $K = 10,000$ with a code dimension of $d_c = 896$. The vector-quantization loss weight is set to $\lambda_{\mathrm{VQ}} = 0.25$, and the commitment loss weight is $\beta = 0.1$.

We optimize the model using the AdamW [29] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.01. The training spans 3 epochs on the 1M rendered corpus with a peak learning rate of $1\mathrm{e}{-4}$ and a global batch size of 64. To prevent early codebook collapse, we employ a two-branch curriculum where the continuous-branch annealing factor $\alpha(e)$ is linearly decayed from 1 to 0 over the course of the first epoch.

C.3 Latent Language Model Training Configurations

The language models are trained with a maximum sequence length of $L_{\max} = 256$ text and latent tokens. The learning rates for all stages follow a cosine-annealing schedule with a 0.03 warmup ratio. The specific setups for the three stages are as follows:

- **Stage 1 (Latent-Text Alignment):** Only the input/output projectors $(P_{\mathrm{in}}, P_{\mathrm{out}})$ are optimized, while the LLM backbone is completely frozen. The projectors are implemented as single linear layers mapping between the codebook dimension d_c and the backbone hidden dimension d_{lm} (e.g., $d_{\mathrm{lm}} = 4096$ for Qwen3-VL-8B). This stage runs for 3 epochs with a learning rate of $1\mathrm{e}{-4}$ and a global batch size of 64.
- **Stage 2 (Latent Supervised Fine-Tuning):** The LLM/LMM backbone, the codebook embeddings, and the projectors are jointly updated. The auxiliary decoder $\mathbf{D}_{\mathrm{ocr}}$ is trained in parallel, but its gradients are stopped at the backbone and codebook interface. This stage runs for 3 epochs with a per-GPU batch size of 8 (resulting in a global batch size of 64), a learning rate of $2\mathrm{e}{-5}$, and an AdamW weight decay of 0.01.
- **Stage 3 (Latent Reinforcement Learning):** The codebook and $\mathbf{D}_{\mathrm{ocr}}$ are strictly frozen, while the backbone and projectors are updated via GRPO. This stage is trained for 1 epoch with a learning rate of $1\mathrm{e}{-6}$. We sample $G = 32$ rollouts per prompt using a temperature of 1.0. The GRPO algorithm performs 4 PPO inner epochs per batch. The reward coefficients are set to $\lambda_{\mathrm{proc}} = 0.25$ for the process alignment reward and $\lambda_{\mathrm{fmt}} = 0.1$ for the format verification reward, while the KL-divergence penalty coefficient against the reference policy is set to $\beta_{\mathrm{KL}} = 0.001$.

D Additional Main Results

To expand the evaluation of our proposed method, this section presents extended experimental results on smaller model scales and detailed scalability analyses.

D.1 Extended Baseline Comparisons at 1B and 2B Scales

Table 5 reports the detailed performance of **DLR** against all baselines on the lightweight LLaMA-3.2-1B and Qwen3-VL-2B-Instruct architectures. The results confirm that the advantages of discrete latent reasoning persist even in capacity-constrained regimes. On the **LLaMA-3.2-1B** backbone, **DLR** achieves 51.3% average accuracy across the four grade-school benchmarks. It clearly outperforms the strongest continuous-latent baseline, ReGuLaR (45.6%), which also leverages rendered CoT but relies on a continuous variational prior. Furthermore, **DLR** maintains a compact latent footprint (averaging ~ 5.2 tokens), offering a favorable trade-off between computational efficiency and reasoning accuracy compared to explicit CoT-SFT (113.0 tokens).

This advantage transfers directly to the **Qwen3-VL-2B-Instruct** multimodal backbone. **DLR** outperforms the visually aligned continuous baseline, RoT (37.0% average accuracy), establishing a new

Table 5: Main results on four grade-school reasoning benchmarks with LLaMA-3.2-1B (LLM) and Qwen3-VL-2B-Instruct (LMM). * indicate the Pass@1 significantly outperform the second-best latent reasoning method ($p < 0.05$). The results of baseline methods are from CoLaR or RoT.

	GSM8K-Aug		GSM-Hard		SVAMP		MultiArith		Average	
Method	Acc.	#L	Acc.	#L	Acc.	#L	Acc.	#L	Acc.	#L
LLaMA-3.2-1B										
Direct-SFT	$13.9_{\pm.3}$	$0.0_{\pm.0}$	$4.1_{\pm.4}$	$0.0_{\pm.0}$	$41.3_{\pm.3}$	$0.0_{\pm.0}$	$32.2_{\pm.2}$	$0.0_{\pm.0}$	22.9	0.0
CoT-SFT	$54.1_{\pm.4}$	$123.2_{\pm.9}$	$15.6_{\pm.3}$	$197.5_{\pm.7}$	$66.7_{\pm.1}$	$61.8_{\pm1.1}$	$99.3_{\pm.2}$	$69.4_{\pm.6}$	58.9	113.0
iCoT	$19.8_{\pm.2}$	$0.0_{\pm.0}$	$3.9_{\pm.2}$	$0.0_{\pm.0}$	$36.4_{\pm.5}$	$0.0_{\pm.0}$	$38.2_{\pm.7}$	$0.0_{\pm.0}$	24.6	0.0
Coconut	$20.5_{\pm.7}$	$6.0_{\pm.0}$	$4.9_{\pm.3}$	$6.0_{\pm.0}$	$39.8_{\pm.7}$	$6.0_{\pm.0}$	$41.4_{\pm.7}$	$6.0_{\pm.0}$	26.6	6.0
CODI	$13.3_{\pm.6}$	$6.0_{\pm.0}$	$3.0_{\pm.2}$	$6.0_{\pm.0}$	$21.7_{\pm.7}$	$6.0_{\pm.0}$	$19.2_{\pm.8}$	$6.0_{\pm.0}$	14.3	6.0
CoLaR	$26.6_{\pm.2}$	$5.6_{\pm.0}$	$6.2_{\pm.1}$	$7.0_{\pm.1}$	$47.1_{\pm.3}$	$3.0_{\pm.0}$	$87.0_{\pm.2}$	$3.2_{\pm.0}$	41.7	4.7
ReGuLaR	$34.9_{\pm.3}$	$3.7_{\pm.2}$	$8.3_{\pm.1}$	$4.1_{\pm.5}$	$50.1_{\pm.4}$	$2.0_{\pm.2}$	$89.2_{\pm.3}$	$2.3_{\pm.3}$	45.6	3.0
DLR (Full)	$\mathbf{43.4}_{\pm.4}$	$6.4_{\pm.3}$	$\mathbf{11.0}_{\pm.1}$	$6.6_{\pm.2}$	$\mathbf{57.3}_{\pm.2}$	$3.5_{\pm.1}$	$\mathbf{93.7}_{\pm.3}$	$4.1_{\pm.2}$	**51.3**	5.2
Qwen3-VL-2B-Instruct										
Direct-SFT	$15.6_{\pm.3}$	$0.0_{\pm.0}$	$4.7_{\pm.2}$	$0.0_{\pm.0}$	$52.3_{\pm.3}$	$0.0_{\pm.0}$	$41.7_{\pm.3}$	$0.0_{\pm.0}$	28.6	0.0
CoT-SFT	$59.7_{\pm.4}$	$131.4_{\pm1.6}$	$33.1_{\pm.3}$	$207.2_{\pm1.7}$	$67.3_{\pm.3}$	$63.4_{\pm.8}$	$95.0_{\pm.4}$	$68.0_{\pm.7}$	63.8	117.5
Coconut	$13.4_{\pm.3}$	$6.0_{\pm.0}$	$4.2_{\pm.3}$	$6.0_{\pm.0}$	$41.5_{\pm.2}$	$6.0_{\pm.0}$	$52.9_{\pm.3}$	$6.0_{\pm.0}$	28.0	6.0
CODI	$9.3_{\pm.2}$	$6.0_{\pm.0}$	$2.4_{\pm.1}$	$6.0_{\pm.0}$	$11.3_{\pm.2}$	$6.0_{\pm.0}$	$17.5_{\pm.1}$	$6.0_{\pm.0}$	10.1	6.0
CoLaR	$28.3_{\pm.2}$	$5.7_{\pm.2}$	$8.2_{\pm.3}$	$6.5_{\pm.2}$	$49.3_{\pm.3}$	$3.2_{\pm.0}$	$79.4_{\pm.3}$	$3.4_{\pm.1}$	41.3	4.7
RoT	$23.3_{\pm.3}$	$32.0_{\pm.0}$	$8.6_{\pm.2}$	$32.0_{\pm.0}$	$53.7_{\pm.4}$	$32.0_{\pm.0}$	$62.2_{\pm.4}$	$32.0_{\pm.0}$	37.0	32.0
DLR (Full)	$\mathbf{53.4}_{\pm.3}$	$6.0_{\pm.2}$	$\mathbf{15.4}_{\pm.3}$	$6.3_{\pm.2}$	$\mathbf{67.7}_{\pm.4}$	$3.8_{\pm.4}$	$\mathbf{94.4}_{\pm.5}$	$4.2_{\pm.3}$	**57.7**	5.1

state-of-the-art for lightweight latent reasoning models. Specifically, RoT requires a fixed length of 32 continuous tokens per reasoning step, whereas **DLR** achieves higher accuracy while using a dynamically generated, compressed discrete latent sequence. This confirms that explicitly discretizing the latent space provides a stronger semantic anchor for lightweight models than purely continuous visual alignment.

D.2 Scalability Analysis Across Model Sizes

To investigate how different latent reasoning methods benefit from increased model capacity, we analyze the scalability across 1B, 3B, and 8B parameter scales on the four grade-school-level benchmarks (Fig. 5).

As illustrated by the scaling trajectories, while continuous latent methods such as Coconut and CoLaR show initial improvements from 1B to 3B, they face diminishing returns and plateau at the 8B scale. ReGuLaR achieves higher baseline performance but similarly suffers from scaling bottlenecks, particularly on out-of-domain tasks like GSM-Hard and SVAMP.

In contrast, **DLR** displays a steeper and more consistent scaling curve, outperforming ReGuLaR across all datasets and scale points. The performance gap between **DLR** and the continuous baselines *widens* as the model size increases on the GSM8K-Aug, GSM-Hard, and SVAMP datasets. This consistent scaling behavior highlights a core advantage of our discrete method: unlike continuous hidden states that may suffer from representation saturation or drift over long reasoning chains, our discrete visual-semantic tokens provide stable structural boundaries. As the backbone model grows larger, it can better exploit these discrete symbolic anchors to model advanced logical dependencies, unlocking deeper reasoning capabilities without being bottlenecked by the continuous latent space.

E Additional Ablation Study

E.1 Latent Codebook Construction

We study how two codebook design choices affect reasoning performance: the visual encoder and the compression ratio. As reported in Tab. 6, all results are evaluated after Stage-2 SFT to isolate the structural effect of the codebook from RL refinement. First, replacing the document-pretrained DeepEncoder V2 with the native Qwen3-VL ViT reduces average accuracy by 3.0 points (from 57.3%

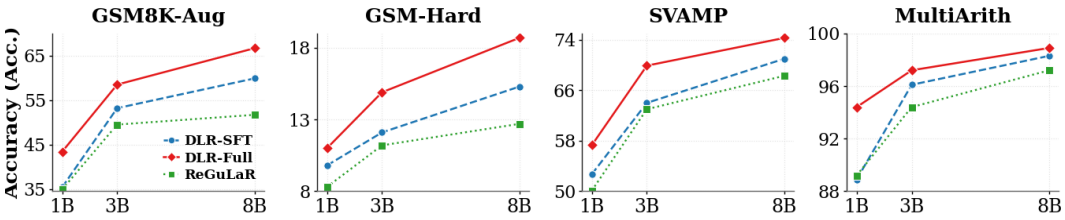


Figure 5: Scalability analysis across varying model sizes.

Table 6: Ablation on the **visual encoder** and **compression setting** used for latent-codebook construction. We compare the DeepEncoder V2 from DeepSeek-OCR2 against the native ViT of Qwen3-VL, each evaluated under a *default* and a *high-compression* setting. All variants share the same Qwen3-VL-4B backbone and are reported after **Stage-2 latent SFT (no RL)**. **#L** denotes the average length of the discrete latent-token trajectory; a smaller **#L** reflects a higher compression ratio.

Encoder	GSM8K-Aug		GSM-Hard		SVAMP		MultiArith		Average	
	Acc.	#L	Acc.	#L	Acc.	#L	Acc.	#L	Acc.	#L
DeepEncoder V2 (DeepSeek-OCR2)										
Default	$48.0_{\pm.4}$	6.3	$14.7_{\pm.3}$	6.8	$68.7_{\pm.2}$	3.6	$97.8_{\pm.3}$	4.2	57.3	5.2
High compression	$42.6_{\pm.2}$	3.2	$11.5_{\pm.3}$	3.5	$59.7_{\pm.3}$	2.1	$92.8_{\pm.2}$	2.2	51.7	2.8
Qwen3-VL ViT (native)										
Default	$44.2_{\pm.5}$	31.2	$12.5_{\pm.1}$	31.5	$66.3_{\pm.2}$	31.4	$93.9_{\pm.4}$	31.5	54.2	31.4
High compression	$39.8_{\pm.2}$	16.3	$11.4_{\pm.2}$	16.7	$62.7_{\pm.0}$	15.2	$89.4_{\pm.3}$	15.8	50.8	16.0

to 54.2%). Furthermore, the native ViT increases the average latent trajectory length (#L) from 5.2 to 31.4 tokens. This confirms that DeepEncoder V2 provides a superior and more compact basis for latent quantization. Second, we compare a *default* setting with a *high-compression* setting (half of the rendered font size). The default setting preserves most of the causal reasoning signal, while the high-compression setting trades accuracy (-5.6 points) for an even shorter latent footprint (2.8 tokens), indicating that **DLR** degrades gracefully rather than failing completely under high compression. Finally, since DeepEncoder V2 is pretrained on roughly square document crops and its patchification is defined for square canvases, we use square rendering throughout **DLR**; exploring non-square rendering would require retraining the encoder and is left to future work.

E.2 Latent Decoding

In this section, we ablate how the model transitions from latent reasoning to the textual answer. Because the decoding strategy directly shapes RL rollouts, we run this study on the *Stage-3 RL checkpoint*. We compare the default *dynamic* decoding, where the model autonomously emits the end-of-latent special token and switches to textual generation, with a family of *forced-early-stop* variants that terminate the latent phase after $n \in \{4, 6, 8, 10\}$ latent steps regardless of content. As reported in Tab. 7, dynamic decoding achieves the best average accuracy (63.2%). Forced early stopping at n=4 causes the largest performance drop (-12.4 points on average) because arithmetic problems require more than four discrete latent steps to accumulate intermediate quantities. Conversely, stopping at n=10 recovers most of the accuracy (60.5%) at a roughly 6% latent-length reduction (from 5.4 to 5.1 tokens). These results confirm that the model has learned to self-regulate the latent length in a semantically meaningful way, and that the learned dynamic termination outperforms fixed-budget generation.

F Interpretablity Analysis

To further evaluate the interpretability of the DLR method, we present additional latent-to-text decoding examples on out-of-domain (OOD) benchmarks (GSM-Hard, SVAMP, MultiArith) and

Table 7: Ablation on latent decoding with Qwen3-VL-4B-Instruct, reported on the **Stage-3 RL checkpoint**. Dynamic decoding matches the main result; forced-early-stop variants truncate the latent phase at a fixed number of steps.

	GSM8K-Aug		GSM-Hard		SVAMP		MultiArith		Average	
Decoding Strategy	Acc.	#L	Acc.	#L	Acc.	#L	Acc.	#L	Acc.	#L
Dynamic (auto text/latent switch)	**63.3**$_{\pm.3}$	6.5	**18.3**$_{\pm.4}$	7.0	**72.7**$_{\pm.3}$	3.8	**98.3**$_{\pm.2}$	4.3	**63.2**	5.4
Forced early stop, n=4	43.9$_{\pm.2}$	4.0	12.5$_{\pm.1}$	4.0	63.3$_{\pm.3}$	3.2	83.4$_{\pm.2}$	3.4	50.8	3.7
Forced early stop, n=6	52.8$_{\pm.1}$	5.3	13.7$_{\pm.3}$	5.4	66.7$_{\pm.2}$	3.5	90.0$_{\pm.3}$	3.7	55.8	4.5
Forced early stop, n=8	56.9$_{\pm.2}$	5.8	16.1$_{\pm.0}$	5.8	67.3$_{\pm.3}$	3.6	94.4$_{\pm.1}$	4.0	58.7	4.8
Forced early stop, n=10	60.2$_{\pm.4}$	6.2	16.3$_{\pm.2}$	6.2	70.3$_{\pm.1}$	3.7	95.0$_{\pm.2}$	4.2	60.5	5.1

GSM-Hard

Question: Carlos is planting a lemon tree. The tree will cost $90 to plant. Each year it will grow 2256855 lemons, which he can sell for $1.5 each. It costs $3 a year to water and feed the tree. How many years will it take before he starts earning money on the lemon tree?

Decoded CoT: Each lemon $1.5 per week. The total earnings of the lemon is 90*$1.5 = $135. The total year earnings of the lemon is $135*12.

Answer: The answer is: 1.

SVAMP

Question: We ordered 9 pizzas. Each pizza has 10 slices. If there are 2 of us How many slices of pizza does each of us get if distributed equally?

Decoded CoT: The total number of pizza slices is 10 * 9 = 90 slices. The number of slices eaten is 90. So, there were 90 / 2 = 45 people at the party.

Answer: The answer is: 45.

MultiArith

Question: Tom had 5 books. If he sold 4 of them and used the money he earned to buy 38 new books, how many books would Tom have?

Decoded CoT: Tom has 5 books. So, Tom has 5 - 4 = 1 more books. Therefore, Sarah has 1 + 38 = 39 books.

Answer: The answer is: 39.

MATH-500

Question: Convert the point $(0,3)$ in rectangular coordinates to polar coordinates. Enter your answer in the form $(r,\theta),$ where $r > 0$ and $0 \le \theta < 2 \pi.$

Decoded CoT: We can write the original point (0,3), r = 3. This means that. We can write 3. Hence, the original point (0,3), r=3, since θ=π/2. Then the point, and the original point, the only angle. The angle is π/2, and the original point is (3, π/2), which simplifies to solve this is.

Answer: The answer is: (3, π/2).

Figure 6: Case study of latent-to-text decoding on GSM-Hard, SVAMP, MultiArith, and MATH-500 datasets. Best viewed by zoom in.

the high-school-level MATH-500 benchmark. As shown in Fig. 6, the auxiliary decoder translates dense latent trajectories into interpretable chains of thought. An analysis of these decoded trajectories reveals several properties of how the model performs reasoning in the latent space.

Preservation of Core Mathematical Logic. Across all benchmarks, the decoded chains of thought confirm that the latent states capture the correct underlying mathematical operations required to solve the problems. For instance, in the **SVAMP** case, the decoded text correctly formulates the steps $10 \times 9 = 90$ and $90/2 = 45$. Similarly, in the **MultiArith** case, the model correctly executes $5 - 4 = 1$ and $1 + 38 = 39$. For the more advanced **MATH-500** case, the latent states accurately derive the polar coordinates $r = 3$ and $\theta = \pi/2$. This confirms that the discrete latent tokens function as a reliable causal carrier of the computation graph.

Semantic Drift and Superficial Artifacts. While the mathematical reasoning is accurate, the explicitly decoded text often contains minor semantic hallucinations or syntactic artifacts. We categorize these into three distinct phenomena:

- **Entity and Unit Swapping:** The decoder sometimes substitutes subjects or units that do not affect the mathematical outcome. In the **MultiArith** case, the subject abruptly changes from "Tom" to "Sarah" in the final step. In the **SVAMP** case, the decoded text concludes with "45 people at the party" instead of "45 slices".
- **Syntactic Stuttering:** As seen in the **MATH-500** case, the decoded text exhibits repetitive phrasing (*e.g.*, "This means that. We can write 3... Then the point, and the original point...").
- **Numerical Reconstruction Noise:** In the **GSM-Hard** case, where the problem involves large perturbed numbers (*e.g.*, 2256855 lemons), the decoded text hallucinates smaller, misaligned numbers (*e.g.*, $90 * \$1.5 = \$135$). Nevertheless, the latent trajectory outputs the correct final answer ("1").

Discussion. These observations suggest a clear conclusion regarding the nature of continuous/discrete latent reasoning. Because the latent tokens are optimized purely for downstream reasoning accuracy and computational efficiency, rather than linguistic fluency, the latent space naturally abstracts away trivial textual details (such as names, exact object units, or grammatical structure). The auxiliary decoder, acting as a probing window, reconstructs the "skeleton" of the reasoning process. The presence of semantic noise alongside accurate final answers indicates that **DLR** efficiently compresses the causal mathematical logic without being constrained by the verbosity of natural language generation.

G Limitation and Broader Impacts

G.1 Limitations

While **Discrete Latent Reasoning (DLR)** demonstrates strong performance and interpretability, there are limitations in its current implementation. First, the data curation process requires rendering explicit textual chains of thought into images to extract visually compressed features. This introduces computational overhead during dataset preparation and codebook training. However, this overhead is confined to the offline training stage and does not affect the inference efficiency of the final language model. Second, the size of the latent codebook (K) and the quantization hyperparameters may require tuning when adapting the method to different domains (*e.g.*, specialized coding or multilingual tasks). Finally, the interpretability of the latent trajectories relies on the reconstruction quality of the auxiliary OCR decoder; if the decoder is inadequately trained, the latent states remain functional for the LLM's internal reasoning but may decode into less fluent interpretable text.

G.2 Broader Impacts.

The development of efficient latent reasoning models carries positive societal impacts. By compressing verbose text generation into dense discrete tokens, **DLR** reduces the inference time, memory bandwidth, and computational cost required for complex problem-solving. This efficiency enables the deployment of advanced reasoning systems on resource-constrained edge devices, reducing the energy consumption and carbon footprint associated with large language model inference.