RuPLaR : Efficient Latent Compression of LLM Reasoning Chains with Rule-Based Priors From Multi-Step to One-Step

Anonymous ACL submission

Abstract

The Chain-of-Thought (CoT) paradigm, while enhancing the interpretability of Large Language Models (LLMs), is constrained by the inefficiencies and expressive limits of natural language. Latent Chain-of-Thought (latent CoT) reasoning, which operates in a continuous latent space, offers a promising alternative but faces challenges from structural complexities in existing multi-step or multi-model paradigms, such as error propagation and coordination overhead. In this paper, we introduce **One-Model One-Step**, a novel compression framework for **La**tent **R**easoning with **Ru**le-Based **P**riors(**RuPLaR**) to address this challenge. Our method trains an LLM to autonomously generate latent reasoning tokens in a single forward pass, guided by rule-based prior probability distributions, thereby eliminating cascaded processes and inter-model dependencies. To ensure reasoning quality, we design a multi-objective loss function that enforces answer consistency via cross-entropy, aligns soft tokens with a rule-based prior via KL-divergence (the Soft Thinking constraint), and problem-thought semantic alignment constraint that operates in the representation space. Extensive experiments show that our compression framework not only improves accuracy by 10.9% over existing latent CoT methods but also achieves this with minimal token usage, underscoring its effectiveness and extensibility.

1 Introduction

The Chain-of-Thought (CoT) paradigm (Wei et al., 2022), which elicits Large Language Models (LLMs) to articulate step-by-step reasoning in natural language, has significantly enhanced the interpretability and performance of LLMs on complex tasks (Guo et al., 2025; Chen et al., 2025a,b). By externalizing intermediate thought processes, CoT provides a transparent window into the model's decision-making and enables human oversight.

However, the very medium of natural language imposes fundamental constraints on the reasoning capacity of LLMs. The primary limitation is expressive redundancy: many tokens essential for grammar (e.g., "therefore", "of") are functionally unnecessary for reasoning. This leads to inflated token usage and slower inference without a commensurate gain in reasoning quality (Feng et al., 2025; Sui et al., 2025). A second constraint is the semantic bottleneck: human cognition often involves abstract, continuous, or multi-concept representations that transcend discrete linguistic symbols and are difficult to verbalize precisely (Wittgenstein, 1921).

These inherent limitations have motivated a shift from explicit language chains towards **Latent Chain-of-Thought(latent CoT)** reasoning, where computation occurs within a continuous latent space (Zhang et al., 2025a). Latent CoT models replace discrete tokens with high-density, high-dimensional latent representations, which serve as a more natural and efficient substrate for computational reasoning. Nonetheless, this emerging paradigm introduces new core challenges stemming from the unobservable nature of the reasoning process. This opacity creates significant training and coordination difficulties, making direct supervision hard to implement (Xu et al., 2025a; Ruan et al., 2025).

To address these inquiries, research on latent CoT reasoning has undergone rapid and vigorous development. As illustrated in Figure 1, existing methodologies can be broadly classified into three paradigms according to their architectural and procedural designs: *Multi-Model Multi-Step*, *Multi-Model One-Step*, and *One-Model Multi-Step*. The first paradigm (Xu et al., 2025a; Wang et al., 2025b; He et al., 2025; Xu et al., 2025b) employs a cascaded framework in which an assistant model generates soft reasoning traces, which are subsequently utilized by a LLM to produce the final answer. The

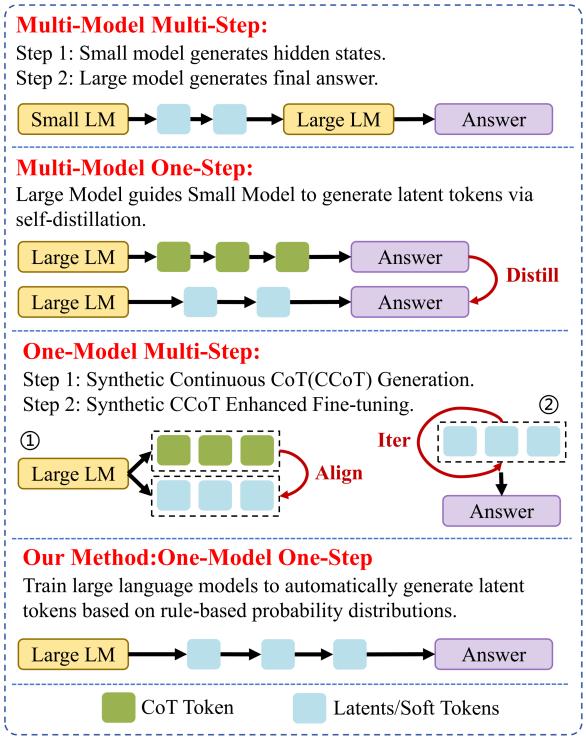


Figure 1: The latent reasoning paradigms of existing approaches and our method. Note that multiple latent reasoning paradigms exist in existing methods; only the most typical paradigm is selected as representative.

second paradigm (Shen et al., 2025; Kuzina et al., 2025) utilizes distillation techniques to condense discrete explicit CoT reasoning into dense soft tokens. The third paradigm (Wu et al., 2025; Tan et al., 2025; Liu et al., 2025) encompasses diverse methodological strategies. Among these, one representative approach (Wang et al., 2025a) involves aligning soft tokens with explicit reasoning chains, followed by iterative refinement of these soft tokens.

While these paradigms have demonstrated promise, they are inherently limited by their structural complexities. The multi-step approaches, both within and across models, often suffer from error propagation, where inaccuracies in early stages compromise the final outcome. Meanwhile, multi-model frameworks frequently face challenges in achieving tight and efficient collaboration between components, leading to suboptimal information flow and increased computational overhead.

Inspired by the above insights, this paper introduces a novel compression paradigm termed *One-Model One-Step*, which effectively reduces token sequence length while preserving state-of-the-art performance. Our method trains a large language model to autonomously and directly generate latent reasoning tokens in a single step, guided by rule-based probability distributions. This integrated approach eliminates the need for cascaded steps or inter-model coordination, thereby mitigating error propagation and fostering a more cohesive and efficient reasoning process within a unified model. To ensure the quality of the latent CoT reasoning, we employ a multi-objective joint loss function for directional optimization, comprising three key components. The first is the **Answer Consistency Constraint**: Implemented as a standard cross-entropy loss, this component ensures the accuracy and coherence of the final output. The second is the **Soft Thinking Paradigm Constraint** (Soft Thinking Paradigm introduced in Section 2): Formulated as a Kullback–Leibler (KL) divergence loss, this term minimizes the divergence between the rule-based prior probability distribution and the distribution represented by each generated soft token. The third is the **Problem-Thought Semantic Constraint**: Also implemented as a KL divergence loss, this term aligns the semantic distribution of generated thinking tokens with that of the input question token, ensuring that the latent reasoning path remains semantically grounded in the original problem context. To prove the effectiveness of our model, we evaluate the proposed model on four benchmark datasets: GSM8k-Aug, GSM-Hard, SVAMP and MultiArith. Extensive experiments show that it achieves state-of-the-art performance with minimal reasoning length and demonstrates the effectiveness of our model for latent reasoning. Code: anonymized for review.

In summary, our main contribution are four-fold:

- To our knowledge, we are the first to train LLMs using prior probability distributions constructed based on certain rules as a supervision signal. This effectively addresses the noise present in the prior probability distribution generated through training.

- We propose a novel *One-Model One-Step* training framework as a variant of the trainable Soft Thinking paradigm. This framework reduces systemic complexity, computational overhead, and inference latency compared to *Multi-Step* or *Multi-Model* alternatives.

- We devise a focused KL-divergence loss that aligns generated soft tokens with Rule-Based

Priors through selective computation on semantically critical tokens. This selective focus mechanism maintains semantic fidelity while granting flexibility across the full vocabulary space.

- Extensive experiments validate the superiority of our method, which not only achieves a 10.9% accuracy gain over existing latent reasoning approaches but also attains this result with the minimum token consumption.

2 Related Works

Our work focuses on latent reasoning. We categorize it into three primary groups based on the origin of the latent representation: the model's internal hidden states, weighted embeddings of existing vocabulary tokens, and newly introduced special vectors.

Hidden State. COCONUT (Hao et al., 2025) pioneered using the final hidden state as input for latent reasoning, though it suffers from spatial inconsistency between hidden states and token embeddings. CODI (Shen et al., 2025) addresses this by incorporating an MLP for mapping and knowledge distillation. PCCOT (Wu et al., 2025) further improves efficiency via Jacobi iteration, while KaVa (Kuzina et al., 2025) leverages KV cache distillation. An alternative approach employs auxiliary models to generate latent thoughts (Xu et al., 2025a; Zhang et al., 2025a; Wang et al., 2025b). In contrast, our method directly operates on the input token embedding space to resolve spatial inconsistency.

Weighted Embedding. Soft Thinking (Zhang et al., 2025b) and MoI (Zhuang et al., 2025) combine token probability distributions with embeddings in a training-free manner. Subsequent work enables training through reinforcement learning (Butt et al., 2025; Zheng and Lee, 2025; Yue et al., 2025). CoLaR (Tan et al., 2025) introduces latent heads to predict compressed embeddings. Our method extends the Soft Thinking paradigm into a trainable framework that significantly reduces token usage while enabling multi-path exploration.

Special Vector. DART (Jiang et al., 2025) and SynAdapt (Wang et al., 2025a) employ dedicated vector parameters through knowledge distillation and synthetic CoT generation. Other approaches insert latent tokens directly into input (Sun et al., 2025) or use complex learned states (Liu et al., 2025; Zheng et al., 2025). For representation learning, Token Assorted (Su et al., 2025) uses VQ-VAE for quantization, while CoCoMix (Tack et al., 2025) employs sparse autoencoders. ReGuLaR (Wang et al., 2026) formulates latent reasoning within a Variational Auto-Encoding (VAE) framework

3 Problem Definition and Notations

This section formalizes the mathematical reasoning task and introduces our notation. The task is defined on a dataset $\mathcal{D}$, where each instance comprises a question $\mathbf{t}_q = (t_q^1, \ldots, t_q^{L_q})$, a reasoning chain $\mathbf{t}_r = (t_r^1, \ldots, t_r^{L_r})$, and an answer $\mathbf{t}_a = (t_a^1, \ldots, t_a^{L_a})$, with L_q, L_r, L_a denoting their respective token lengths. A representative example entry would be: *"Question: Out of 600 employees in a company, 30% got promoted while 10% received bonus. How many employees did not get either a promotion or a bonus?"*, *"Reasoning chain:* `«600*30/100=180» «600*10/100=60» «180+60=240» «600-240=360»`*"*, and *"Answer: 15"*.

In conventional multi-step reasoning, the intermediate chain t_r is often long and token-inefficient, slowing down inference and increasing memory overhead. To overcome this, we propose **reasoning compression**—mapping lengthy reasoning sequences into a compact latent space while preserving semantic fidelity. At the core of this approach is the notion of soft tokens, defined as follows:

Soft Token Representation. A soft token $z \in \mathbb{R}^d$ is defined as a linear combination of the model's vocabulary embeddings. Formally,

$$z = \sum_{i=1}^{|V|} \pi_i e_i, \tag{1}$$

where $e_i \in \mathbb{R}^d$ is the embedding of the i-th token in the vocabulary, and the coefficients $\pi_i \in \mathbb{R}$ control the mixture. This ensures z stays consistent with the model's foundational semantic structure, integrating information from multiple tokens in a principled way.

4 Method

To address the challenge of lengthy reasoning sequences, we introduce a compression-based framework with *One-Model One-Step* that maps multi-step reasoning into a compact latent space, thereby enhancing the efficiency of LLMs. This approach is designed to fulfill three primary objectives: (1)

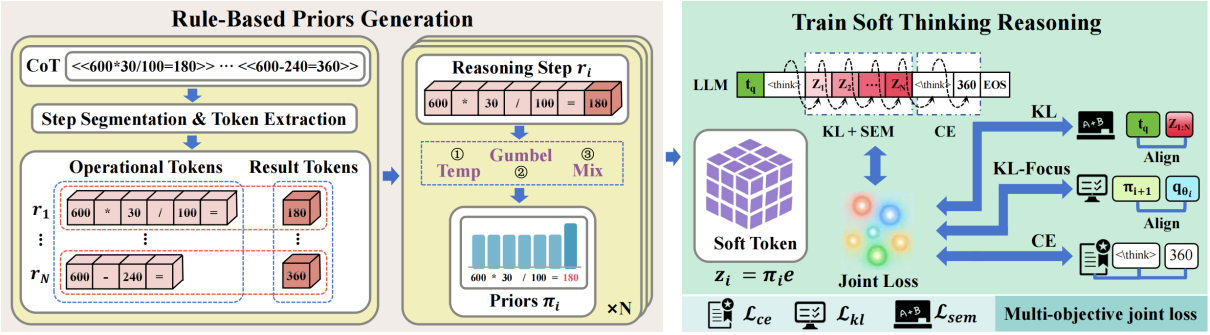


Figure 2: Overview of our model. Our model is trained in two distinct phases: (a) generating initial soft-token priors via rule-based methods, in the data processing stage, and (b) training the LLM to generate soft thinking tokens autonomously. The Rule-based Priors Generation component (left) presents three methodological variants (indicated by dashed borders) for constructing target distributions. The red rounded rectangle represents the output of Step Segmentation, while the blue rounded rectangle represents the output of Token Extraction.

to encode reasoning tokens into dense latent representations and enable the model to interpret these compressed states; (2) to autoregressively predict subsequent latent states and decide when to halt reasoning; and (3) to facilitate latent reasoning exploration and exploitation.

Achieving these goals necessitates first generating soft tokens (**Soft Token Representation in Section 3**) to serve as supervision signals for model training. This is accomplished in a two-stage process: First, in the data processing stage, a prior distribution is constructed using rule-based methods to establish ground-truth latent labels. Second, the model is trained in a manner analogous to explicit reasoning—using these latent labels—to generate latent tokens and perform latent inference. The overall training framework is illustrated in Figure 2.

4.1 Rule-Based Priors Generation

To train the model for latent reasoning, we first need to establish a target distribution for latent states—termed soft token priors—that encodes compressed semantic information of reasoning steps. As direct supervision from human-annotated latent tokens is unavailable, we generate Rule-Based Priors to serve as pseudo-labels through a three-stage process: (1) segmentation of explicit reasoning chains into computational steps, (2) token extraction from each step, and (3) differentiable prior construction. This approach transforms discrete reasoning chains into continuous distributions while preserving essential arithmetic semantics.

To transform the explicit reasoning chain $\mathbf{t}_r$ into a differentiable learning signal, we decompose it into individual computational steps. Formally, given a reasoning step r composed of tokens such as «`20*1.20=24`», we construct a prior distribution $\pi \in \mathbb{R}^{|V|}$ over the vocabulary V, where π is a probability vector (i.e., $\pi_{v_i} \geq 0$ and $\sum_i^{|V|} \pi_{v_i} = 1$). The distribution is designed to concentrate probability mass on semantically relevant tokens while maintaining differentiability for gradient-based learning.

Step Segmentation and Token Extraction. We parse each reasoning chain into discrete computational steps using structural patterns that delineate intermediate reasoning units (e.g., delimited segments like «...»). Unlike fixed-length segmentation approaches (Tan et al., 2025), our method adapts dynamically to the complexity of each reasoning chain, allowing variable numbers of steps per problem. For each extracted step r_i, we identify two fundamental token categories:

- **Operational Tokens:** Numerical values, arithmetic operators, and mathematical symbols that constitute the computational expression. For instance, in the step «`20*1.20=24`», these include `20`, `*`, `1`, `.`, `20`, and `=`.

- **Result Tokens:** The numerical output produced by evaluating the step's expression. In the same example, this corresponds to `24`.

This adaptive tokenization scheme preserves the essential arithmetic logic of multi-step reasoning while accommodating natural variations in problem complexity. By supporting dynamic segmentation, our approach provides a flexible yet structured foundation for subsequent soft token representation learning.

Prior Construction Methods. Following token extraction, we transform the discrete tokens into continuous probability distributions suitable for gradient-based learning. The core challenge lies in balancing semantic specificity with training stability—ensuring the prior concentrates on reasoning-relevant tokens while remaining numerically robust for optimization. We address this through three complementary approaches, each offering distinct trade-offs between focus, smoothness, and differentiability:

(1) **Temperature-based Prior Construction.** We construct the prior distribution π through a temperature-scaled categorical distribution defined over the vocabulary V. Formally,

$$p_{\text{Temp}} = \pi = p(\cdot \mid r_i) = \text{Softmax}\left(\frac{\boldsymbol{\ell}}{\tau}\right), \tag{2}$$

where $\boldsymbol{\ell} \in \mathbb{R}^{|V|}$ is a logit vector. To encode the relevance of extracted tokens, we initialize $\ell_v = -\infty$ for all tokens v not appearing in the reasoning step. For tokens in the operational set $\mathcal{K} = \{v_1, \ldots, v_m\}$, we set $\ell_v = \beta_{\text{op}}$, while for the result token v_{res}, we assign a higher value $\ell_{v_{\text{res}}} = \beta_{\text{res}}$ with $\beta_{\text{res}} > \beta_{\text{op}}$. The temperature hyperparameter $\tau > 0$ governs the entropy of the distribution: as $\tau \to 0$, the distribution approaches a deterministic focus on the highest-logit tokens.

(2) **Gumbel-Softmax Prior for Differentiable Sampling.** For scenarios requiring stochastic latent variables, we employ Gumbel-Softmax relaxation to enable differentiable categorical sampling:

$$p_{\text{Gumbel}} = \text{GumbelSoftmax}(\boldsymbol{\ell}, \tau, \text{hard} = False), \tag{3}$$

where the operation is defined as:

$$p_i = \frac{\exp((\ell_i + g_i)/\tau)}{\sum_{j=1}^{|V|} \exp((\ell_j + g_j)/\tau)}, \quad g_i \sim \text{Gumbel}(0, 1). \tag{4}$$

The parameter hard selects between soft relaxation (training) and discrete approximation (inference), maintaining gradient flow while supporting stochastic reasoning representations.

(3) **Mixture Prior for Balanced Token Emphasis.** To explicitly balance attention between computational process and final outcome, we design a mixture prior:

$$p_{\text{mix}} = \lambda \cdot p_{\text{uniform}} + (1 - \lambda) \cdot p_{\text{result}}, \tag{5}$$

where p_{uniform} distributes probability uniformly over operational tokens $\mathcal{K}$:

$$p_{\text{uniform}}(v) = \begin{cases} \frac{1}{|\mathcal{K}|}, & \text{if } v \in \mathcal{K} \\ 0, & \text{otherwise} \end{cases}, \tag{6}$$

and p_{result} concentrates on the result token t_{result}:

$$p_{\text{result}}(v) = \mathbb{I}[v = t_{\text{result}}]. \tag{7}$$

The mixing coefficient $\lambda \in [0, 1]$ provides explicit control over the trade-off between modeling the computational procedure ($\lambda \to 1$) and emphasizing the final answer ($\lambda \to 0$).

These three methods offer different trade-offs between semantic focus, training stability, and computational properties, allowing flexible adaptation to varying training requirements while maintaining the core objective of encoding reasoning semantics into differentiable priors. More details about Rule-Based Priors Construction Pipeline in Appendix G.

4.2 Train Soft Thinking Reasoning

The training of our soft-thinking reasoning model employs a multi-objective optimization framework that simultaneously ensures answer accuracy, prior distribution alignment, and semantic consistency. As illustrated in Figure 2 (right), our approach directly optimizes the latent reasoning process through three complementary loss components, each addressing a specific aspect of the reasoning task.

Answer Consistency Constraint. To ensure the model produces correct final answers, we employ a standard cross-entropy loss on the output sequence:

$$\mathcal{L}_{\text{ce}} = -\frac{1}{L_a} \sum_{i=1}^{L_a} \log p(t_a \mid t_q, \mathbf{z}), \tag{8}$$

where y_i is the i-th token of the ground truth answer, x is the input question, and $\mathbf{z}$ represents the generated soft-thinking tokens. This component guarantees that the compressed reasoning process ultimately leads to accurate solutions.

Soft Thinking Paradigm Constraint. We introduce a focused KL-divergence loss that selectively aligns soft tokens with Rule-Based Priors on semantically critical tokens. Unlike standard approaches that enforce alignment across the entire vocabulary, our method concentrates computation on the most relevant tokens, reducing complexity from $O(|V|)$ to $O(k)$ while filtering gradient noise and preserving semantic specificity. This selective mechanism maintains fidelity to reasoning-essential vocabulary items, offering a principled alternative to exhaustive alignment strategies.

Given a generated soft token distribution $q_\theta(z_{\text{dist}} \mid \cdot)$ from the LLM and its corresponding Rule-Based Priors $p_{\text{prior}}(z_{\text{dist}})$, we compute:

$$\mathcal{L}_{\text{kl}} = \frac{1}{N} \sum_{j=1}^{N} \sum_{v \in \mathcal{T}_j} p_{\text{prior}}(v) \left[\log p_{\text{prior}}(v) - \log q_\theta(v) \right], \tag{9}$$

where $\mathcal{T}_j$ denotes the set of top-k tokens in $p_{\text{prior}}(z_{\text{dist}})$ with probability above threshold δ. This selective alignment ensures that key reasoning elements are preserved in the latent representation.

Problem-Thought Semantic Constraint. While the prior alignment constraint ensures mathematical correctness at the distribution level, we further introduce a semantic coherence constraint that operates in the representation space to enforce logical flow and information consistency throughout the reasoning chain. This constraint addresses a fundamental aspect of coherent reasoning: problem-thought semantic alignment, which ensures that each reasoning step remains relevant to the original question.

Formally, let $\mathbf{h}_q \in \mathbb{R}^d$ denote the hidden representation of the question $\mathbf{t}_q$ at its final token position, obtained from the LLM's encoder. For each generated soft thinking token $\mathbf{z}_i$ (defined in Section 3) in the latent reasoning path of length N, we obtain its corresponding hidden representation $\mathbf{h}_{z_i} \in \mathbb{R}^d$ for $i = 1, \ldots, N$. To enforce semantic alignment between each thinking token and the original question, we minimize the Kullback–Leibler (KL) divergence between their probability distributions:

$$\mathcal{L}_{\text{sem}} = D_{\text{KL}} \left(\text{Softmax}(\mathbf{h}_q) \parallel \text{Softmax}(\mathbf{h}_z) \right) \tag{10}$$

where $D_{\text{KL}}(\cdot \parallel \cdot)$ denotes the KL divergence, and the softmax function transforms the hidden representations into probability distributions over the feature space. This formulation anchors each latent reasoning step to the original problem context, preventing the model from drifting into irrelevant reasoning paths. By minimizing this divergence, we ensure that the distribution represented by each soft thinking token remains semantically grounded in the question.

Unified Training Objective. The complete training objective combines all three constraints with appropriate weighting coefficients:

$$\mathcal{L}_{\text{total}} = \alpha_{\text{ce}} \mathcal{L}_{\text{ce}} + \alpha_{\text{kl}} \mathcal{L}_{\text{kl}} + \alpha_{\text{sem}} \mathcal{L}_{\text{sem}}, \tag{11}$$

where $\alpha_{\text{ce}}, \alpha_{\text{kl}}, \alpha_{\text{sem}} > 0$ are hyperparameters that balance the relative importance of answer correctness, prior alignment, and problem-thought semantic alignment, respectively.

5 Experiments

5.1 Experimental setup

Datasets. We train and evaluate our method on GSM8k-Aug (Deng et al., 2023), an augmented variant of GSM8k (Cobbe et al., 2021) for grade-school math, containing 385K training and 1K test samples. For out-of-domain evaluation, we use: (1) GSM-Hard (Gao et al., 2023) (1K test samples with larger numbers); (2) SVAMP (Patel et al., 2021) (1K samples); and (3) MultiArith (Roy and Roth, 2015) (600 samples). Following (Tan et al., 2025), we report Accuracy (Acc.) and reasoning chain length (# L), along with their ratio (Acc./# L) to jointly measure performance and efficiency.

Baseline methods. We compare our method against several representative baselines: (1) **CoT** (Wei et al., 2022) is fine-tuned on complete reasoning chains and corresponding answers, performing token-level reasoning during inference before generating the final answer. (2) **iCoT** (Deng et al., 2024) internalizes reasoning knowledge by progressively eliminating reasoning steps during training, enabling direct answer prediction without explicit reasoning at inference time. We also compare with three recent methods introduced in the related work: (3) **COCONUT** (Hao et al., 2025), (4) **CODI** (Shen et al., 2025), (5) **CoLaR** (Tan et al., 2025) and (6) **ReGuLaR** (Wang et al., 2026).

Implementation. Unless otherwise specified, all experiments use a frozen Llama-3.2-1B-Instruct (Dubey et al., 2024) backbone with a trainable LoRA module (Hu et al., 2022). Following COCONUT, all methods are initialized with weights from CoT-SFT to accelerate training. More details about experimental have been provided in Appendix A.

5.2 Main Results

We compare our model against five baselines. "Temp", "Gumbel", and "Mix" denote three variants for constructing rule-based priors (Section 4). CoLaR-5 compresses five reasoning tokens into one latent representation. Bold indicates best performance.

As shown in Table 1, RuPLaR consistently outperforms existing methods. On GSM8k-Aug, RuPLaR-Mix achieves 45.8% accuracy—10.9 points above ReGuLaR—with fewer tokens. On

Table 1: Main results and ablation studies on four datasets. #L denotes average latent reasoning steps.

Model	GSM8k-Aug			GSM-Hard			SVAMP			MultiArith			Average		
	Acc.	# L	Acc./# L	Acc.	# L	Acc./# L	Acc.	# L	Acc./# L	Acc.	# L	Acc./# L	Acc.	# L	Acc./# L
iCoT	19.8	0.00	-	3.87	0.00	-	36.4	0.00	-	38.2	0.00	-	24.6	0.00	-
COCONUT	23.1	6.00	3.85	5.49	6.00	0.92	40.7	6.00	6.78	41.1	6.00	6.85	27.6	6.00	4.60
CODI	13.3	6.00	2.22	2.97	6.00	0.50	21.7	6.00	3.62	19.2	6.00	3.20	14.3	6.00	2.38
CoLaR-5	26.8	5.57	4.81	5.87	6.53	0.90	48.4	2.95	16.4	86.4	3.21	26.9	41.7	4.57	9.12
ReGuLaR	34.9	3.69	9.46	8.27	4.12	2.01	50.1	2.02	24.8	89.8	2.04	39.1	45.6	3.03	15.0
RuPLaR-Temp	44.3	**2.97**	14.9	9.86	**2.78**	3.55	**56.8**	**1.50**	**37.9**	89.8	2.04	44.0	50.2	**2.32**	21.6
RuPLaR-Gumbel	45.0	3.03	14.9	10.2	2.88	3.54	54.8	1.56	35.1	93.7	**2.02**	46.4	50.9	2.37	21.5
RuPLaR-Mix	**45.8**	3.00	**15.3**	**10.8**	2.87	**3.76**	54.3	1.59	34.2	**94.8**	2.03	**46.7**	**51.4**	2.37	**21.7**
Rule-Based Prior Variants															
Random Prior	34.1	2.81	12.1	8.26	2.65	3.12	48.6	1.42	34.2	81.0	1.98	40.9	43.0	2.22	19.4
Uniform Prior	35.1	3.03	11.6	8.04	3.24	2.48	47.3	1.56	30.3	72.2	2.03	35.6	40.7	2.47	16.5
Learned Prior	38.9	3.05	12.8	9.78	2.88	3.40	45.2	1.66	27.2	82.3	2.07	39.8	44.0	2.42	18.2
Prior Strength Variants															
Weak ($\alpha_{\text{kl}} = 0.3$)	40.9	2.83	14.5	9.33	2.62	3.56	52.5	1.43	36.7	91.5	1.99	46.0	48.6	2.22	21.9
Strong ($\alpha_{\text{kl}} = 2.0$)	41.3	2.98	13.9	9.47	2.87	3.30	51.4	1.57	32.7	89.2	2.01	44.4	47.8	2.36	20.3
Loss Component Variants															
w/o KL-Focus	39.6	2.97	13.3	9.40	2.76	3.41	48.9	1.60	30.6	89.5	2.06	43.4	46.9	2.35	20.0
w/o Problem-Thought	39.1	2.95	13.3	9.70	2.76	3.51	51.0	1.56	32.7	88.8	2.04	43.5	47.3	2.33	20.3

GSM-Hard, it reaches 10.8% accuracy, surpassing ReGuLaR by 2.53 points. On SVAMP, RuPLaR-Temp attains 56.8% accuracy—6.7 points higher than ReGuLaR—using only 1.50 tokens. On MultiArith, RuPLaR-Mix achieves 94.8% accuracy. Overall, RuPLaR-Mix leads with 51.4% average accuracy, a 5.8% absolute improvement over ReGuLaR, while using significantly fewer tokens.

Mixture Prior Superiority. RuPLaR-Mix achieves the best overall performance among the three variants. Its mixture prior balances operational and result tokens via coefficient λ, providing more stable supervision than temperature-based or Gumbel approaches. This design captures both computational procedures ($\lambda \to 1$) and answer emphasis ($\lambda \to 0$), yielding robust representations. RuPLaR-Gumbel achieves competitive results (50.9%) but slightly underperforms the mixture variant, suggesting that deterministic priors avoid sampling instability. RuPLaR-Temp excels on SVAMP (56.8%) with minimal token usage (2.32 average) but slightly trails in overall accuracy.

Efficiency-Effectiveness Trade-off. RuPLaR achieves superior accuracy with minimal token usage. RuPLaR-Mix attains 21.7 Acc./# L on average, substantially outperforming ReGuLaR (15.0), COCONUT (4.60), and CoLaR-5 (9.12). This validates our core motivation: compressing multi-step reasoning with rule-based guidance enhances reasoning quality while reducing computational overhead. The fact that RuPLaR achieves this without cascaded models or inter-model dependencies underscores the effectiveness of our One-Model One-Step compression framework.

5.3 Ablation Study

To comprehensively validate the effectiveness of our proposed Rule-Based Priors, we conduct extensive ablation experiments across all four mathematical reasoning datasets. Table 1 presents the complete results, from which we draw several key insights.

Rule-based Prior Variants. We first examine the fundamental contribution of our rule-based prior by comparing alternative configurations within the Gumbel-Softmax framework. Replacing rule-based scores with randomly assigned token importance (**Random Prior**) yields 43.0% average accuracy—an 8.4-point drop from the full model's 51.4%—with consistent degradation across datasets. Assigning equal importance to all tokens (**Uniform Prior**) performs even worse at 40.7% (a 10.7-point decline), confirming that differentiated token importance is crucial; treating all tokens uniformly dilutes structural information in reasoning chains. A data-driven prior based on token position statistics (**Learned Prior**) recovers some performance (44.0%) but still lags 7.4 points behind, particularly on challenging datasets (GSM8k-Aug: 38.9% vs. 45.8%; MultiArith: 82.3% vs. 94.8%). These ablations validate that our rule-based prior provides essential inductive biases—prioritizing result tokens over operational tokens—that random, uniform, or purely statistical alternatives cannot replicate.

Prior Strength Variants. We further examine the impact of the KL divergence weight α_{kl}. A weaker prior ($\alpha_{\text{kl}} = 0.3$) achieves 48.6% accuracy with 2.22 steps, while a stronger prior ($\alpha_{\text{kl}} = 2.0$) yields 47.8% accuracy with 2.36 steps. The full model with $\alpha_{\text{kl}} = 1.0$ strikes the optimal balance at 51.4% accuracy and 2.37 steps. This consistent pattern across datasets demonstrates that moderate prior strength is optimal: too weak provides insufficient guidance, while too strong over-constrains the model and limits its flexibility.

Loss Component Variants. Finally, we ablate two key loss components. Removing the focused KL-divergence (**w/o KL-Focus**) drops accuracy from 51.4% to 46.9%—a 4.5-point decrease—with consistent degradation, confirming its role in preserving semantic fidelity by filtering gradient noise. Removing the Problem-Thought Semantic Constraint (**w/o Problem-Thought**) yields 47.3% accuracy—a 4.1-point drop—with declines across all datasets, demonstrating its function in grounding reasoning steps in the problem context. Together, these results validate that both components play complementary roles: KL-Focus ensures distributional correctness, while Problem-Thought maintains semantic coherence.

5.4 Comparative Analysis with CoLaR Under K=2 Compression

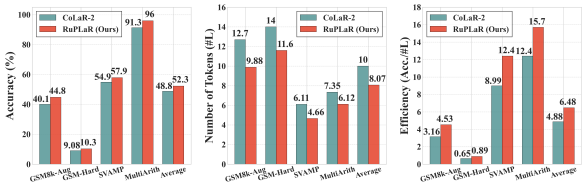


Figure 3: Comparison with CoLaR-2 under identical compression ratio (k=2), showing accuracy, token usage, and efficiency across four datasets.

To ensure a fair comparison with CoLaR under the same compression setting, we evaluate our RuPLaR framework using k=2 compression, where every two reasoning tokens are compressed into a single latent token. Under this configuration, our method disables the Token Extraction component—operational tokens and result tokens are treated uniformly with equal probability assignment—enabling a direct and equitable comparison with CoLaR-2.

As shown in Figure 3, RuPLaR consistently outperforms CoLaR-2 across all datasets. On average, RuPLaR achieves 52.3% accuracy versus CoLaR-2's 48.8%, a substantial 3.5% absolute improvement, while using fewer tokens (8.07 vs. 10.0). This efficiency gain is further reflected in the Acc./#L metric, where RuPLaR attains 6.48 compared to CoLaR-2's 4.88—a 32.8% relative improvement.These results demonstrate that even without Token Extraction, RuPLaR's Rule-Based Priors and multi-objective optimization enable superior latent representations under identical compression constraints. Moreover, they suggest that beyond simply increasing the number of soft tokens to boost performance, the Token Extraction component offers an additional orthogonal avenue for enhancement, which our full model leverages to achieve even greater gains.

6 Conclusion

We introduce One-Model One-Step, a compression framework for RuPLaR that trains a single LLM to generate latent reasoning tokens in one forward pass guided by Rule-Based Priors, eliminating cascaded processes and inter-model dependencies. A multi-objective loss function combines answer consistency, focused KL-divergence for selective token alignment, and a problem-thought constraint to ground reasoning in the question context. Experiments on four mathematical benchmarks show RuPLaR achieves 45.8% accuracy on GSM8k-Aug, outperforming ReGuLaR by 10.9 points with fewer tokens. Collectively, these findings establish RuPLaR as an effective and efficient paradigm for latent reasoning, offering a robust foundation for future research in this direction.

Limitations

Despite the promising results achieved by RuPLaR, several limitations warrant discussion. First, the three prior construction methods—temperature-based, Gumbel-Softmax, and mixture—require careful hyperparameter tuning. As shown in our analysis, the choice of β_{res}, β_{op}, λ, and τ significantly impacts performance, with suboptimal settings leading to substantial accuracy degradation. This sensitivity may hinder deployment in new tasks without extensive validation. Second, our experiments focus primarily on mathematical reasoning benchmarks; the effectiveness of RuPLaR on broader NLP tasks requiring diverse reasoning types—such as commonsense, temporal, or causal reasoning—remains to be validated.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Natasha Butt, Ariel Kwiatkowski, Ismail Labiad, Julia Kempe, and Yann Ollivier. 2025. Soft tokens, hard truths. *arXiv preprint arXiv:2509.19170*.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. 2025a. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*.

Xinghao Chen, Anhao Zhao, Heming Xia, Xuan Lu, Hanlin Wang, Yanjun Chen, Wei Zhang, Jian Wang, Wenjie Li, and Xiaoyu Shen. 2025b. Reasoning beyond language: A comprehensive survey on latent chain-of-thought reasoning. *arXiv preprint arXiv:2505.16782*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Yuntian Deng, Yejin Choi, and Stuart Shieber. 2024. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. 2025. Efficient reasoning models: A survey. *arXiv preprint arXiv:2504.10903*.

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*, pages 10764–10799. PMLR.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. 2025. Training large language models to reason in a continuous latent space. In *Workshop on Reasoning and Planning for Large Language Models*.

Yinhan He, Wendy Zheng, Yaochen Zhu, Zaiyi Zheng, Lin Su, Sriram Vasudevan, Qi Guo, Liangjie Hong, and Jundong Li. 2025. Semcot: Accelerating chain-of-thought reasoning through semantically-aligned implicit tokens. *arXiv preprint arXiv:2510.24940*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Nan Jiang, Ziming Wu, De-Chuan Zhan, Fuming Lai, and Shaobing Lian. 2025. Dart: Distilling autoregressive reasoning to silent thought. *arXiv preprint arXiv:2506.11752*.

Anna Kuzina, Maciej Pioro, Paul N Whatmough, and Babak Ehteshami Bejnordi. 2025. Kava: Latent reasoning via compressed kv-cache distillation. *arXiv preprint arXiv:2510.02312*.

Jiayu Liu, Zhenya Huang, Anya Sims, Enhong Chen, Yee Whye Teh, and Ning Miao. 2025. Marcos: Deep thinking by markov chain of continuous thoughts. *arXiv preprint arXiv:2509.25020*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1743–1752.

Yangjun Ruan, Neil Band, Chris J Maddison, and Tatsunori Hashimoto. 2025. Reasoning to learn from latent thoughts. *arXiv preprint arXiv:2503.18866*.

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. *arXiv preprint arXiv:2502.21074*.

DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. 2025. Token assorted: Mixing latent and text tokens for improved language model reasoning. *arXiv preprint arXiv:2502.03275*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Yuchang Sun, Yanxi Chen, Yaliang Li, and Bolin Ding. 2025. Enhancing latent computation in transformers with latent tokens. *arXiv preprint arXiv:2505.12629*.

Jihoon Tack, Jack Lanchantin, Jane Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason Weston, and Xian Li. 2025. Llm pretraining with continuous concepts. *arXiv preprint arXiv:2502.08524*.

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. *arXiv preprint arXiv:2505.16552*.

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

Jianwei Wang, Ziming Wu, Fuming Lai, Shaobing Lian, and Ziqian Zeng. 2025a. Synadapt: Learning adaptive reasoning in large language models via synthetic continuous chain-of-thought. *arXiv preprint arXiv:2508.00574*.

Jiaqi Wang, Binquan Ji, Haibo Luo, Yiyang Qi, Ruiting Li, Huiyan Wang, Yuantao Han, Cangyi Yang, Feiliang Ren, and 1 others. 2025b. Lta-thinker: Latent thought-augmented training framework for large language models on complex reasoning. *arXiv preprint arXiv:2509.12875*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Ludwig Wittgenstein. 1921. Tractatus logico-philosophicus (annalen der naturphilosophie). *Ostwald (traducción española (1957), Tierno Galván, Enrique, Madrid, Revista de Occidente, reediciones 1973, 1975, 1979, Madrid, Alianza Editorial)*.

Haoyi Wu, Zhihao Teng, and Kewei Tu. 2025. Parallel continuous chain-of-thought with jacobi iteration. *arXiv preprint arXiv:2506.18582*.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025a. SoftCoT: Soft chain-of-thought for efficient reasoning with LLMs. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23336–23351, Vienna, Austria. Association for Computational Linguistics.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. 2025b. Softcot++: Test-time scaling with soft chain-of-thought reasoning. *arXiv preprint arXiv:2505.11484*.

Zhenrui Yue, Bowen Jin, Huimin Zeng, Honglei Zhuang, Zhen Qin, Jinsung Yoon, Lanyu Shang, Jiawei Han, and Dong Wang. 2025. Hybrid latent reasoning via reinforcement learning. *arXiv preprint arXiv:2505.18454*.

Guibin Zhang, Fanci Meng, Guancheng Wan, Zherui Li, Kun Wang, Zhenfei Yin, Lei Bai, and Shuicheng Yan. 2025a. Latentevolve: Self-evolving test-time scaling in latent space. *arXiv preprint arXiv:2509.24771*.

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

Haoyu Zheng, Zhuonan Wang, Yuqian Yuan, Tianwei Lin, Wenqiao Zhang, Zheqi Lv, Juncheng Li, Siliang Tang, Yueting Zhuang, and Hongyang He. 2025. Fast thinking for large language models. *arXiv preprint arXiv:2509.23633*.

Zhi Zheng and Wee Sun Lee. 2025. Soft-grpo: Surpassing discrete-token llm reinforcement learning via gumbel-reparameterized soft-thinking policy optimization. *arXiv preprint arXiv:2511.06411*.

Yufan Zhuang, Liyuan Liu, Chandan Singh, Jingbo Shang, and Jianfeng Gao. 2025. Text generation beyond discrete token sampling. *arXiv preprint arXiv:2505.14827*.

A More Experimental Details

A.1 Datasets

Following prior work (Tan et al., 2025), we train and evaluate our method primarily on the GSM8K-Aug dataset (Deng et al., 2023), and further assess its generalization on three out-of-domain mathematical reasoning datasets: GSM-Hard (Gao et al., 2023), SVAMP (Patel et al., 2021), and MultiArith (Roy and Roth, 2015). Detailed descriptions of these datasets are provided below:

- **GSM8K-Aug** (Deng et al., 2023): An augmented version of the original GSM8K dataset (Cobbe et al., 2021), constructed by prompting GPT-4 (Achiam et al., 2023) to expand the training set to 385K samples. Notably, it removes natural language descriptions from the reasoning process, formalizing each reasoning step as a mathematical expression.

- **GSM-Hard** (Gao et al., 2023): A harder variant of GSM8K, created by replacing numbers in the original test set with larger, less common values. This dataset serves as an out-of-domain benchmark in our experiments.

- **SVAMP** (Patel et al., 2021): Consists of 1,000 elementary-level math word problems designed to test model robustness on fundamental mathematical tasks. It is used as an out-of-domain dataset in our evaluation.

- **MultiArith** (Roy and Roth, 2015): Contains 600 multi-step arithmetic problems aimed at assessing a model's capacity for multi-step reasoning. This dataset also serves as an out-of-domain benchmark.

A.2 Implementation Details

We fine-tune all models using LoRA (Hu et al., 2022) with rank $r = 32$ and scaling factor $\alpha = 64$, applying adaptations to all linear layers in the attention and feed-forward modules—including `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`—as well as the token embedding layer `embed_tokens`. To ensure fair comparison, all models are trained for up to 10 epochs, and we select the checkpoint with the highest validation accuracy as the final model. Training batch sizes are set to 32, 10, 4, and 24 for Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct, Llama-3.1-8B-Instruct, and DeepSeek-R1-Distill-Qwen-1.5B, respectively. For the loss functions, we set $\beta_{\text{res}} = 2.8$, $\beta_{\text{op}} = 2.0$, and $\tau = 0.5$ in Equation 2; $\delta = 1 \times 10^{-2}$ in Equation 9; and $\alpha_{\text{ce}} = \alpha_{\text{kl}} = \alpha_{\text{sem}} = 1$ in Equation 11. During inference, we use a temperature of 0.8 and top-p of 0.95 for generation. The random seed is fixed to 777 for reproducibility.

B Scalability Analysis Across Model Sizes and Families

To assess the generalizability of our RuPLaR framework, we evaluate its performance across different backbone architectures and model scales. Specifically, we experiment with the LLaMA-3.2 family (1B and 3B parameters) and the DeepSeek-R1-Distill-Qwen-1.5B model. Table 2 summarizes the results.

Scaling Within the LLaMA Family. Increasing model capacity from 1B to 3B parameters consistently improves reasoning performance. The LLaMA-3B backbone achieves higher average accuracy (53.7% vs. 51.4%) and superior results on individual datasets, particularly on GSM8k-Aug (49.8% vs. 45.8%) and MultiArith (96.7% vs. 94.8%). This gain, however, comes with a modest increase in token usage (2.51 vs. 2.37), yielding comparable efficiency (21.4 vs. 21.7 Acc./#L). These results indicate that larger models better leverage latent reasoning while maintaining similar token efficiency.

Cross-Architecture Generalization. When applied to the DeepSeek-R1-Distill-Qwen-1.5B architecture, RuPLaR achieves 46.6% average accuracy with 2.39 tokens, demonstrating strong cross-architecture transferability. Although this is lower than the LLaMA-1B counterpart (51.4%), the performance remains competitive given the architectural differences. Notably, on SVAMP, DeepSeek achieves 52.2% accuracy—approaching LLaMA-1B's 54.3%—with reasonable efficiency (19.5 Acc./#L). These findings suggest that RuPLaR's rule-based priors and multi-objective optimization are architecture-agnostic and can be effectively applied across diverse model families.

C Analysis of Hyperparameters in Gumbel-Softmax Prior Construction

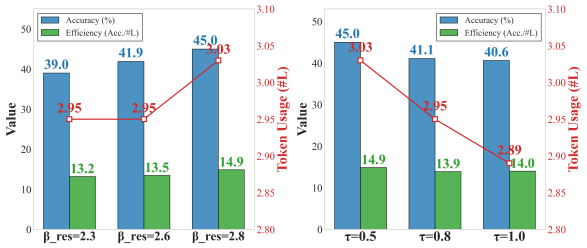


Figure 4: Hyperparameter analysis on the GSM8k-Aug dataset, showing the impact of result token weight β_{res} and temperature τ on accuracy, token usage, and efficiency.

We conduct a hyperparameter analysis for the Gumbel-Softmax prior construction on the GSM8k-Aug dataset, examining the impact of both the result token weight β_{res} and temperature τ on model performance. As illustrated in Figure 4, both hyperparameters significantly influence reasoning quality and efficiency.

Effect of Result Token Weight β_{res}. Increasing β_{res} from 2.3 to 2.8, while keeping $\beta_{\text{op}} = 2.0$ and $\tau = 0.5$ fixed, yields substantial performance gains. Accuracy rises from 39.0% to 45.0%—a notable 6.0% absolute improvement—demonstrating that stronger emphasis on result tokens significantly enhances reasoning quality. Although token usage increases slightly from 2.95 to 3.03, the efficiency metric Acc./#L improves markedly from 13.2 to 14.9, indicating that the performance gains outweigh the modest increase in token consumption. This confirms that appropriately prioritizing result tokens during prior construction plays a critical role in guiding the model toward more accurate reasoning.

Effect of Temperature τ. With $\beta_{\text{res}} = 2.8$ and $\beta_{\text{op}} = 2.0$ fixed, temperature τ exhibits a clear in-

Table 2: Scalability analysis across different model architectures and sizes. We evaluate LLaMA-3.2 (1B, 3B) and DeepSeek-R1-Distill-Qwen-1.5B as backbone LLMs.

Backbone	GSM8k-Aug			GSM-Hard			SVAMP			MultiArith			Average		
	Acc.	# L	Acc./# L	Acc.	# L	Acc./# L	Acc.	# L	Acc./# L	Acc.	# L	Acc./# L	Acc.	# L	Acc./# L
LLaMA Family															
LLaMA-1B	45.8	3.00	15.3	10.8	2.87	3.76	54.3	1.59	34.2	94.8	2.03	46.7	51.4	2.37	21.7
LLaMA-3B	49.8	3.17	15.7	11.2	3.09	3.62	57.1	1.74	32.8	96.7	2.05	47.1	53.7	2.51	21.4
DeepSeek Family															
DS-1.5B	40.8	3.03	13.5	9.70	2.90	3.34	52.2	1.61	32.4	83.5	2.03	41.1	46.6	2.39	19.5

verse relationship with performance. The lowest temperature ($\tau = 0.5$) achieves the best results: 45.0% accuracy and 14.9 efficiency. Increasing τ to 0.8 leads to a noticeable accuracy drop to 41.1% (a 3.9-point decrease), with token usage slightly decreasing to 2.95. At $\tau = 1.0$, accuracy further declines to 40.6% while token usage reduces to 2.89, though efficiency remains comparable at 14.0. This trend aligns with the theoretical properties of Gumbel-Softmax: as $\tau \to 0$, the distribution approaches a discrete categorical distribution, producing sharper, more deterministic samples that better preserve semantic focus. Conversely, higher temperatures introduce greater smoothing and stochasticity, which, while potentially beneficial for exploration during training, lead to less stable latent representations and diluted semantic signals.

D Analysis of Hyperparameters in Mix Prior Construction

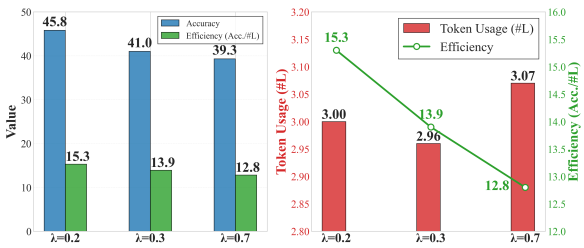


Figure 5: Hyperparameter analysis on the GSM8k-Aug dataset, showing the impact of mixing coefficient λ on accuracy, token usage, and efficiency.

We conduct a hyperparameter analysis for the mixture prior construction on the GSM8k-Aug dataset, focusing on the impact of the mixing coefficient λ, which controls the trade-off between emphasizing operational tokens (via uniform distribution) and the result token (via focused distribution). As shown in Figure 5, we evaluate three settings: $\lambda = 0.2$, $\lambda = 0.3$, and $\lambda = 0.7$.

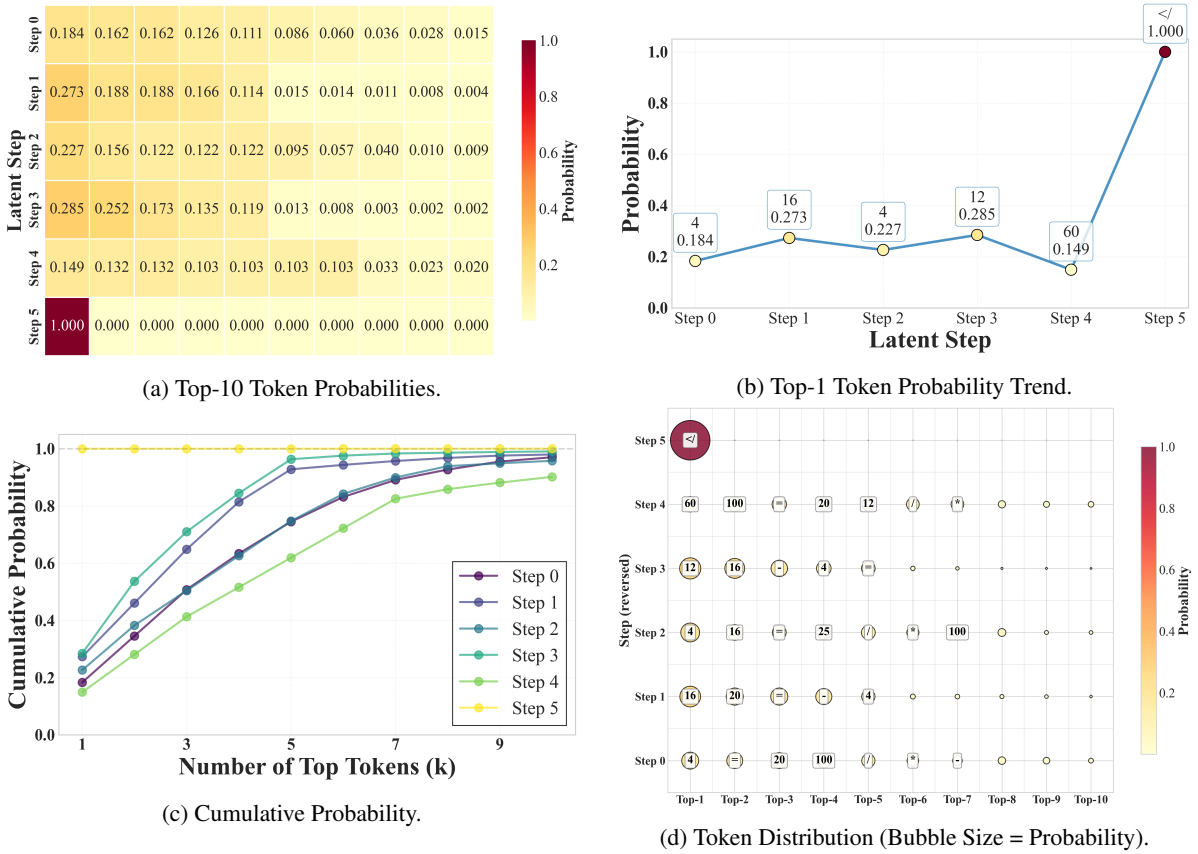
The results demonstrate that a lower λ, which assigns higher weight to the result token ($1 - \lambda = 0.8$), yields the best performance. Specifically, with $\lambda = 0.2$, the model achieves 45.8% accuracy using 3.00 tokens, resulting in the highest efficiency-accuracy ratio of 15.3. Increasing λ to 0.3 leads to a noticeable accuracy drop to 41.0% (a 4.8-point decrease), while token usage slightly decreases to 2.96, indicating that over-emphasizing operational tokens at the expense of the result token harms reasoning quality. At $\lambda = 0.7$, where the prior focuses predominantly on operational tokens, accuracy further declines to 39.3% with increased token usage (3.07), yielding the lowest efficiency ratio of 12.8.

These findings underscore the critical role of the result token in guiding latent reasoning. While operational tokens capture the computational procedure, the result token anchors the final outcome, and a proper balance—favoring the result token—is essential for optimal performance. The mixture prior with $\lambda = 0.2$ achieves the best trade-off, validating our design choice and providing empirical guidance for hyperparameter selection in practice.

E Analysis of Latent Reasoning Dynamics

To gain insight into the latent reasoning process, we visualize the probability distributions of soft tokens for a representative example from GSM8k-Aug, where the model generates six latent steps before producing the final answer. The problem is: *"In a dance class of 20 students, 20% enrolled in contemporary, 25% of remaining in jazz, and rest in hip-hop. What percentage are in hip-hop?"* The explicit reasoning chain comprises five sequential calculations: step 1 computes contemporary enrollment ($20 \times 20/100 = 4$), step 2 calculates remaining students ($20 - 4 = 16$), step 3 determines jazz enrollment ($16 \times 25/100 = 4$), step 4 obtains hip-hop enrollment ($16 - 4 = 12$), and step 5 derives the final percentage ($12/20 \times 100 = 60$).

Figure 6 visualizes the corresponding latent reasoning dynamics. The token probability heatmap (a) shows that while early steps exhibit distributed



(a) Top-10 Token Probabilities.

(b) Top-1 Token Probability Trend.

(c) Cumulative Probability.

(d) Token Distribution (Bubble Size = Probability).

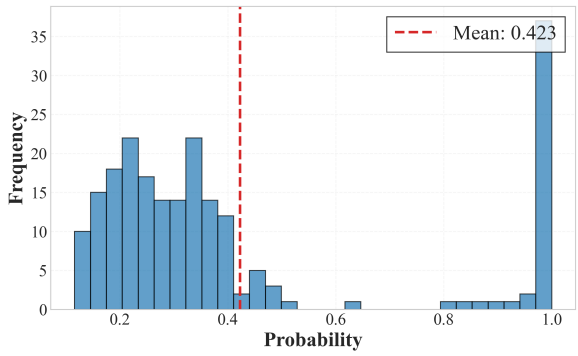
Figure 6: Analysis of latent reasoning dynamics. The visualizations illustrate (a) the probability distribution of top-10 tokens across latent steps, (b) the trend of top-1 token probabilities with corresponding token labels, (c) cumulative probabilities of top-k tokens at each step, and (d) a bubble chart where bubble size and color intensity represent token probability magnitude.

probabilities reflecting exploratory behavior, step 5 assigns near-certain probability (1.0) to the termination token `</think>`, indicating confident reasoning completion. This explore-then-converge pattern is further evidenced by the top-1 probability trend (b), where confidence remains moderate (0.18–0.29) during the first four steps before sharply rising to 0.99 at step 5. The cumulative probability analysis (c) reveals that early steps concentrate probability mass on few tokens, demonstrating focused attention, while the near-vertical line at step 5 confirms deterministic termination.

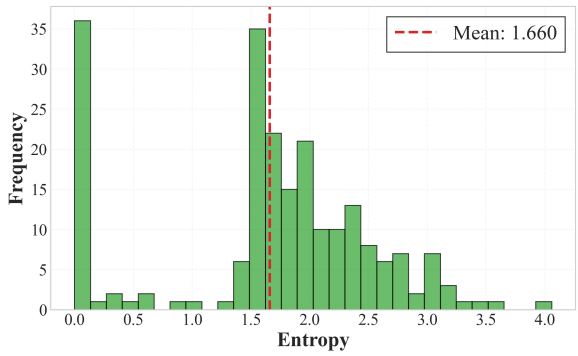
Most notably, the token distribution bubble chart (d) reveals that latent tokens closely mirror the explicit arithmetic progression: step 0 captures the first operation ("4", "20", "100"), step 1 encodes the intermediate result ("16"), step 2 reflects the next calculation ("4", "25"), step 3 represents the subsequent outcome ("12", "16"), step 4 encodes the final answer ("60"), and step 5 terminates with `</think>`. This alignment between latent representations and explicit computational steps demonstrates that RuPLaR successfully distills multi-step arithmetic reasoning into compact latent tokens while preserving logical structure and termination behavior, validating the effectiveness of our Rule-Based Priors and multi-objective optimization approach.

F Batch Statistics Analysis

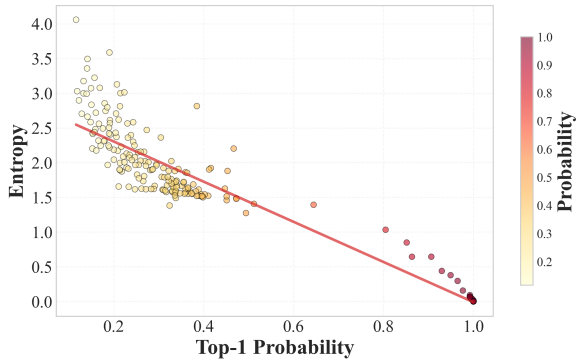
To understand the aggregate behavior of our latent reasoning model, we analyze statistics from 50 randomly sampled GSM8k-Aug instances. Figure 7 reveals key patterns in the latent reasoning process. The top-1 probability distribution (Figure 7a) exhibits a clear bimodal pattern (mean 0.423), with peaks in the moderate range (0.2–0.4) and near 1.0, indicating distinct exploration and termination phases. This pattern is mirrored in the entropy distribution (Figure 7b, mean 1.660), which shows corresponding peaks at near-zero and moderate values. The strong negative correlation between top-1 probability and entropy (Figure 7c) further validates this two-phase dynamic, with points clustering in two



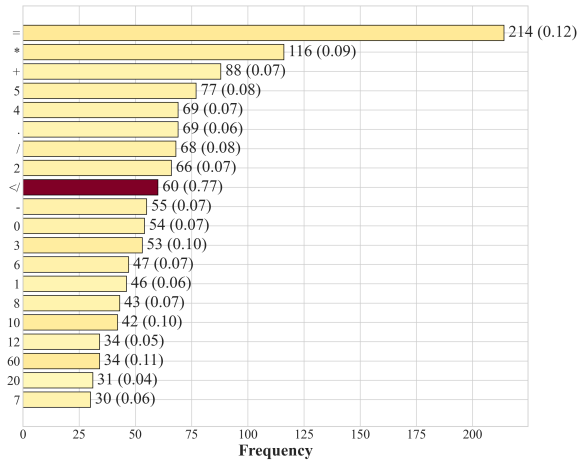
(a) Top-1 Probability Distribution.



(b) Entropy Distribution.



(c) Top-1 Probability vs. Entropy.



(d) Top-20 Most Frequent Tokens.

Figure 7: Batch statistics analysis across 50 GSM8k-Aug instances. The visualizations show (a) distribution of top-1 token probabilities, (b) distribution of entropy values, (c) relationship between top-1 probability and entropy, and (d) frequency of the most common tokens in latent representations.

distinct regions corresponding to exploratory and terminating steps.

Token frequency analysis (Figure 7d) identifies three dominant categories aligned with our Rule-Based Priors design: arithmetic operators (`=`: 214, `*`: 116, `/`: 68, `-`: 55), numerical values (`4`: 69, `2`: 66, `3`: 53, `6`: 47, `20`: 31), and the termination token `</think>` (60). The prevalence of `=` underscores its central role in mathematical expressions, while the diversity of numbers reflects the variety of arithmetic operations encountered in the dataset. Together, these statistics confirm that RuPLaR learns a structured latent reasoning process characterized by clear exploration-termination dynamics and strong alignment with underlying arithmetic semantics.

G Rule-Based Priors Construction Pipeline

This appendix provides a detailed description of the Rule-Based Priors Construction pipeline, which transforms explicit reasoning chains into differentiable prior distributions for training our latent reasoning model. The complete procedure is formalized in Algorithm 1.

G.1 Overview

The pipeline comprises five main steps: (1) segmentation of explicit reasoning chains into computational steps, (2) extraction of operational and result tokens from each step, (3) initialization of logit vectors based on token categories, (4) construction of differentiable prior distributions using one of three methods, and (5) optional selective focus for KL-divergence computation. Each step is designed to preserve essential arithmetic semantics while ensuring differentiability for gradient-based optimization.

G.2 Step-by-Step Description

Step 1: Step Segmentation. Given an explicit reasoning chain $\mathbf{t}_r$, we first parse it into individual computational steps $\{r_i\}_{i=1}^{N}$ using pattern-based

delimiters (e.g., «. . . »). Unlike fixed-length segmentation approaches, our method adapts dynamically to the complexity of each reasoning chain, allowing variable numbers of steps per problem.

Step 2: Token Extraction. For each extracted step r_i, we identify two fundamental token categories:

- **Operational tokens** $\mathcal{K}_i$: numerical values, arithmetic operators, and mathematical symbols that constitute the computational expression.

- **Result token** v^i_{res}: the numerical output produced by evaluating the step's expression.

This adaptive tokenization scheme preserves the essential arithmetic logic while accommodating natural variations in problem complexity.

Step 3: Logit Initialization. We initialize a logit vector $\boldsymbol{\ell} \in \mathbb{R}^{|V|}$ over the vocabulary V, setting $\ell_v = -\infty$ for all tokens not appearing in the step. For operational tokens in $\mathcal{K}_i$, we assign $\ell_v = \beta_{\text{op}}$, and for the result token v^i_{res}, we assign a higher value $\ell_v = \beta_{\text{res}}$ with $\beta_{\text{res}} > \beta_{\text{op}}$.

Step 4: Prior Construction. Based on the chosen method M, we construct differentiable prior distributions:

- **Temperature-based prior** (M = Temp): $\pi_i = \mathrm{Softmax}(\boldsymbol{\ell}/\tau)$, where τ controls distribution entropy.

- **Gumbel-Softmax prior** (M = Gumbel): $\pi_i(v) = \frac{\exp((\ell_v+g_v)/\tau)}{\sum_{u\in V}\exp((\ell_u+g_u)/\tau)}$ with $g_v \sim \mathrm{Gumbel}(0,1)$, enabling differentiable categorical sampling.

- **Mixture prior** (M = Mix): $\pi_i = \lambda \cdot p_{\text{uniform}} + (1-\lambda)\cdot p_{\text{result}}$, where p_{uniform} distributes probability uniformly over $\mathcal{K}_i$ and p_{result} concentrates on v^i_{res}.

Step 5: Selective Focus (Optional). When using focused KL-divergence for training, we identify the top-k tokens $\mathcal{T}_i$ from π_i with probability above threshold δ, retaining only these tokens for subsequent alignment. This reduces computational complexity from $O(|V|)$ to $O(k)$.

G.3 Output

The pipeline produces a sequence of prior distributions $\{\pi_i\}_{i=1}^N$, each corresponding to a reasoning step. These distributions serve as supervision signals for training the latent reasoner, enabling it to learn compressed distributional representations that preserve the symbolic structure and logical flow of the original reasoning process while maintaining differentiability for gradient-based optimization.

Algorithm 1 Rule-Based Priors Construction Pipeline

Require:
- Explicit reasoning chain $\mathbf{t}_r = (t_r^1, \ldots, t_r^{L_r})$
- Vocabulary V, token embeddings $\{e_v\}_{v \in V}$
- Hyperparameters: τ, β_{op}, β_{res}, λ, k, δ
- Prior construction method $M \in \{\text{Temp}, \text{Gumbel}, \text{Mix}\}$

Ensure: Sequence of prior distributions $\{\pi_i\}_{i=1}^N$ over V

Step 1: Step Segmentation
Parse $\mathbf{t}_r$ into computational steps $\{r_i\}_{i=1}^N$ using pattern-based delimiters (e.g., «. . . »)
for $i \leftarrow 1$ to N **do**
    Step 2: Token Extraction
    Identify operational token set $\mathcal{K}_i \subset V$ from r_i
    Identify result token $v_{\text{res}}^i \in V$
    Step 3: Logit Initialization
    Initialize logit vector $\boldsymbol{\ell} \in \mathbb{R}^{|V|}$ with $\ell_v \leftarrow -\infty$ for all $v \in V$
    for each $v \in \mathcal{K}_i$ **do**
        $\ell_v \leftarrow \beta_{\text{op}}$
    end for
    $\ell_{v_{\text{res}}^i} \leftarrow \beta_{\text{res}}$
    Step 4: Prior Construction
    if $M = \text{Temp}$ **then**
        $\pi_i \leftarrow \mathrm{Softmax}(\boldsymbol{\ell}/\tau)$
    else if $M = \text{Gumbel}$ **then**
        Sample $g_v \sim \mathrm{Gumbel}(0, 1)$ for each $v \in V$
        $\pi_i(v) \leftarrow \frac{\exp((\ell_v + g_v)/\tau)}{\sum_{u \in V} \exp((\ell_u + g_u)/\tau)}$
    else if $M = \text{Mix}$ **then**
        $p_{\text{uniform}}(v) \leftarrow \mathbb{I}[v \in \mathcal{K}_i]/|\mathcal{K}_i|$
        $p_{\text{result}}(v) \leftarrow \mathbb{I}[v = v_{\text{res}}^i]$
        $\pi_i \leftarrow \lambda \cdot p_{\text{uniform}} + (1 - \lambda) \cdot p_{\text{result}}$
    end if
    Step 5: Selective Focus (Optional)
    if using focused KL-divergence **then**
        Identify top-k tokens $\mathcal{T}_i$ from π_i with probability $> \delta$
        Retain only $\mathcal{T}_i$ for subsequent alignment
    end if
end for
return $\{\pi_i\}_{i=1}^N$