

Think Silently, Think Fast: Dynamic Latent Compression of LLM Reasoning Chains

Wenhui Tan^{1*}

Jiaze Li²

Jianzhong Ju²

Zhenbo Luo²

Ruihua Song¹✉

Jian Luan²✉

¹Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China

²MiLM Plus, Xiaomi Inc., Beijing, China

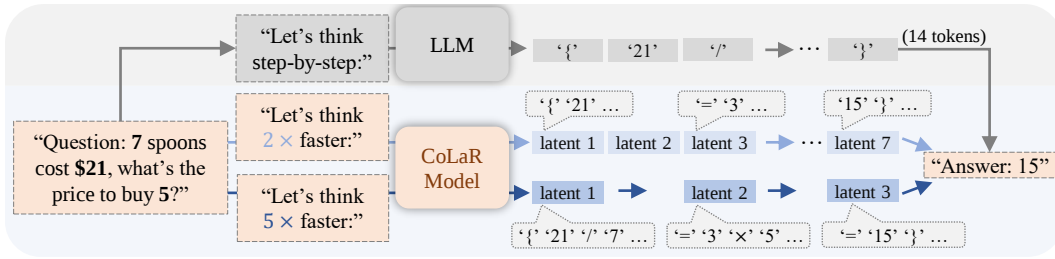


Figure 1: Our proposed Compressed Latent Reasoning Model (CoLaR) performs dynamic-speed reasoning by auto-regressively predicting latent variables, each compressing information from multiple word tokens. Simply prompting to reason faster enables CoLaR to predict more informative latents.

Abstract

Large Language Models (LLMs) achieve superior performance through Chain-of-Thought (CoT) reasoning, but these token-level reasoning chains are computationally expensive and inefficient. In this paper, we introduce Compressed Latent Reasoning (CoLaR), a novel framework that dynamically compresses reasoning processes in latent space through a two-stage training approach. First, during supervised fine-tuning, CoLaR extends beyond next-token prediction by incorporating an auxiliary next compressed embedding prediction objective. This process merges embeddings of consecutive tokens using a compression factor c randomly sampled from a predefined range, and trains a specialized latent head to predict distributions of subsequent compressed embeddings. Second, we enhance CoLaR through reinforcement learning (RL) that leverages the latent head’s non-deterministic nature to explore diverse reasoning paths and exploit more compact ones. This approach enables CoLaR to: i) **perform reasoning at a dense latent level** (i.e., silently), substantially reducing reasoning chain length, and ii) **dynamically adjust reasoning speed** at inference time by simply prompting the desired compression factor. Extensive experiments across four mathematical reasoning datasets demonstrate that CoLaR achieves 14.1% higher accuracy than latent-based baseline methods at comparable compression ratios, and reduces reasoning chain length by 53.3% with only 4.8% performance degradation compared to explicit CoT method. Moreover, when applied to more challenging mathematical reasoning tasks, our RL-enhanced CoLaR demonstrates performance gains of up to 5.4% while dramatically reducing latent reasoning chain length by 82.8%.

✉ Corresponding authors: Ruihua Song (rsong@ruc.edu.cn) and Jian Luan (luanjian@xiaomi.com).
Project page: <https://github.com/xiaomi-research/colar>.

*This Work was performed when Wenhui Tan was visiting Xiaomi as a research intern.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities in mathematical reasoning, particularly when employing Chain-of-Thought (CoT) prompting techniques [29, 31]. Recent advances have further highlighted the potential of this approach when combined with reinforcement learning on extended reasoning sequences [16, 12, 28], revealing significant “aha-moments” in model performance. Despite these advances, a critical limitation persists: generating lengthy reasoning chains are computationally costly, impeding efficiency and scalability. This inefficiency becomes particularly evident in real-world LLM applications, where extended reasoning chains create substantial server load, especially under high-concurrency conditions, underscoring the urgent need for more efficient reasoning methods.

Several approaches have emerged to address this computational challenge [26, 7]. One line of research focuses on enhancing efficiency at the token level, primarily by identifying and skipping less informative tokens [30], prompting models to generate more concise reasoning steps [32, 1], or dynamically terminating reasoning when the model exhibits high confidence in a trial answer [35]. While valuable, these methods continue to operate on sparse token-based representations. A more promising direction explores reasoning within the dense latent space. Initial efforts attempt to “internalize” reasoning knowledge by curriculum learning [5] or knowledge distillation [6]. Some works focus on the potential inside LLMs by looping or skipping some intermediate LLM layers [3, 2, 22] to realize efficient reasoning. Recent innovations have introduced auto-regressive prediction of latent representations for efficient reasoning. Coconut [13] proposes to gradually replace token-level reasoning with latent representations, while CODI [24] employs self-distillation to transfer CoT knowledge into latent reasoning processes. However, these methods primarily utilize **fixed-length reasoning chains**, resulting in suboptimal efficiency and limited adaptability. Furthermore, to the best of our knowledge, all these latent-based methods employ **deterministic latent reasoning processes**, overlooking the potential benefits of exploration-exploitation capability may bring about.

To overcome these limitations, we introduce Compressed Latent Reasoning (CoLaR), a novel framework that dynamically compresses LLM reasoning chains into latent space while preserving exploration-exploitation capabilities. Our approach utilizes an auxiliary *next compressed embedding prediction* task in supervised fine-tuning (SFT) stage. Specifically, at each **training** step, CoLaR first samples a random compression factor $c \in [1, c_{\max}]$ and **merges the embeddings** of c consecutive reasoning tokens using our Embedding Compress module. A Latent Head is then trained to **predict the next compressed embeddings** from the LLM’s output hidden states, which is a fully parallelized process. **During inference**, CoLaR is capable to auto-regressively predict dense and informative latents with the Latent Head, and automatically determine when to terminate the reasoning process with LLM’s Language Head. Rather than predicting deterministic values, the Latent Head outputs a probability distribution that produces diverse reasoning pathways for a same question input. Based on this, we further enhance CoLaR through post-training with Group Relative Policy Optimization (GRPO) reinforcement learning algorithm (**author?**) [23, 37], which enables CoLaR to **explore** correct latent reasoning paths with diverse outputs and **exploit** those shorter ones.

Our extensive evaluations on four grade-school level mathematical reasoning datasets (GSM8k [4], GSM8k-hard [9], SVAMP [20], and MultiArith [21]) demonstrate that CoLaR achieves a **14.1% $\uparrow$** improvement in accuracy compared to state-of-the-art baseline methods at comparable compression ratios. Furthermore, CoLaR reduces reasoning chain length by **53.3% $\downarrow$** with only a **4.8% $\downarrow$** performance degradation relative to explicit CoT. Finally, experiments on a more challenging dataset MATH [14] demonstrates the potential of CoLaR to reinforcement learning, gaining up to **5.36% $\uparrow$** accuracy while reducing the length of reasoning chain significantly by **82.8% $\downarrow$** .

Our main contribution are three-fold:

- We introduce Compressed Latent Reasoning (CoLaR), a novel framework enabling dynamic-speed reasoning by auto-regressively predicting latent variables that encapsulate the compressed semantics of multiple word tokens. This allows for more efficient reasoning by operating in a compressed latent space.
- We design CoLaR with a probabilistic Latent Head and demonstrate the effectiveness of reinforcement learning on latent reasoning. This combination improves performance and reduces the length of reasoning chains by encouraging exploration of diverse reasoning paths and exploitation of the shorter ones.

- Extensive experiments show that CoLaR achieves a 14.1% accuracy improvement over existing latent-based methods. Furthermore, reinforcement learning enhances performance by up to 5.36% while simultaneously reducing reasoning chain dramatically length by 82.8%, demonstrating significant efficiency gains.

2 Related Work

2.1 Explicit LLM reasoning

Recent advances have demonstrated the strong reasoning capabilities of large language models (LLMs). The explicit reasoning approach, exemplified by Chain-of-Thought (CoT) reasoning [29], propose to prompt LLMs to generate intermediate reasoning steps through sequential token prediction before generating answers [36, 31, 27, 39, 17]. Subsequent work demonstrated that reinforcement learning techniques [23, 37, 38] can further improve performance on verifiable reasoning tasks like mathematical problem-solving, revealing an “aha-moment” that significantly boosts model’s performances with longer thinking process [16, 12, 28].

However, the computational cost of processing these lengthy reasoning chains remains a significant bottleneck, motivating research into efficiency optimizations. Current solutions focus on identifying and skipping redundant tokens [30, 35] or encouraging more compact reasoning patterns using mathematical notations and programming language-like expressions [1, 32]. While these methods reduce reasoning chain length, they are fundamentally limited by the sequential token prediction.

2.2 Latent LLM reasoning

Latent reasoning approaches operate in a denser, continuous space, abstracting away from individual word tokens. These methods can be broadly categorized into three directions: knowledge internalization, architectural modifications, and auto-regressive latent reasoning.

The first direction, knowledge internalization, aims to embed reasoning capabilities directly into the model. iCoT-SI [5] attempts to internalize reasoning knowledge by progressively removing explicit reasoning steps during training, while Pause [10] proposes training models to reason within specialized token embeddings.

The second direction exploits the hierarchical structure of transformer layers, with proposals to dynamically skip or repeat layer computations [22, 2, 3, 25, 19]. These methods aim to reduce computational cost by selectively processing different layers.

The third direction, and most relevant to our work, explores auto-regressive latent reasoning [33]. Coconut [13] pioneered this approach by replacing token sampling with hidden state concatenation for breadth-first reasoning, while CODI [24] introduces an auto-regressive latent variable model through self-distillation. However, existing methods like Coconut and CODI are limited by their reliance on fixed-length reasoning chains due to the implicit nature of latent variables. Furthermore, they employ a deterministic approach to auto-regressive latent generation, neglecting the potential for exploration-exploitation strategies to further enhance model performance, particularly within a reinforcement learning framework.

In contrast, CoLaR advances **auto-regressive latent reasoning** by introducing a novel next compressed embedding objective. This allows the model to capture the semantics of multiple word tokens within a single latent variable and reason with dynamic chain lengths, leading to improved efficiency and performance. Moreover, CoLaR achieves significant performance gains and a dramatic reduction in latent reasoning length through reinforcement learning with a probabilistic latent prediction head.

3 Method

In this section, we introduce our task, notations, and our proposed method CoLaR. We focus on mathematical reasoning tasks using a dataset D , which consists of a question $\mathbf{t}_q = t_q^{1:L_q}$, a reasoning chain $\mathbf{t}_r = t_r^{1:L_r}$, and an answer $\mathbf{a}_r = t_a^{1:L_a}$, where L_q , L_r , and L_a denote the respective token lengths. A representative example entry would be: *“Question: A set of 7 spoons costs \$21. If each*

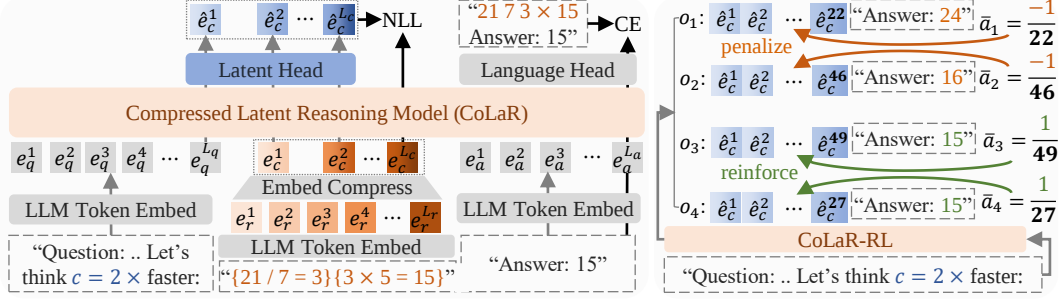


Figure 2: Our proposed method CoLaR consisting an LLM backbone and a Latent Head. During the **SFT stage (left)**, for each training step, CoLaR first compresses embeddings $\mathbf{e}_r$ of the original reasoning chain into compressed embeddings $\mathbf{e}_c$ with a compression factor c randomly selected from the range $[1, c_{max}]$. Then, CoLaR is trained to predict: i) the compressed reasoning embeddings via the Latent Head, and ii) the compressed reasoning tokens and answer tokens through the Language Head. During the **RL stage (right)**, for every question input, CoLaR samples a group of G outputs $o_{1:G}$ consisting of the latent reasoning chain and the predicted answer. We then calculate the relative rewards $a_{1:G}$ for each output, and the rewards are averaged on each token ($\bar{a}_i$), encouraging CoLaR to explore diverse latent reasoning pathways and exploit those more compact ones.

*spoon would be sold separately, how much would 5 spoons cost?", "Reasoning chain: « 21 / 7 = 3 » « 5 * 3 = 15 » <end>", and "Answer: 15".*

Given an LLM backbone $\mathcal{M}$, the input tokens are first mapped to embedding vectors $\mathbf{e}_q = \mathbf{e}_q^{1:L_q}$, $\mathbf{e}_r = \mathbf{e}_r^{1:L_r}$, and $\mathbf{e}_a = \mathbf{e}_a^{1:L_a}$. These embeddings are processed by $\mathcal{M}$ to produce the hidden states of the final layer, denoted $\mathbf{h}_q = \mathbf{h}_q^{1:L_q}$, $\mathbf{h}_r = \mathbf{h}_r^{1:L_r}$, and $\mathbf{h}_a = \mathbf{h}_a^{1:L_a}$. $\mathcal{M}$ then predicts distributions a.k.a. logits of next tokens using a Language Head.

To address this issue of lengthy reasoning chains, we propose compressing reasoning processes into a denser latent space, facilitating more efficient LLM reasoning. This requires our method to i) **compress** reasoning tokens into latent space and **understand** these dense representations, ii) **predict** subsequent dense latent representations and determine when to **terminate** reasoning, and iii) maintain the ability to **explore** diverse latent reasoning paths and **exploit** shorter latent solutions. CoLaR is designed with these three objectives in mind.

3.1 Reasoning token compression and understanding

As illustrated in Figure 2, the input to CoLaR can be represented as $\mathbf{e} = [\mathbf{e}_q, \mathbf{e}_c, \mathbf{e}_a]$, where $[\cdot, \cdot]$ denotes concatenation. Here, $\mathbf{e}_c$ represents the compressed embeddings derived from $\mathbf{e}_r$, the embeddings of the original reasoning steps, and the length of compressed embeddings $L_c = \lceil \frac{L_r}{c} \rceil$. To achieve a dynamic test-time compression factor c , we begin each training step by randomly sampling $c \in [1, r_{max}]$. For every r consecutive reasoning token embeddings $\mathbf{e}_r^{k:k+r}$, the Embedding Compress module generates a compressed embedding $\mathbf{e}_c^k$.

A straightforward approach is to apply mean pooling directly to these embeddings. However, due to the high dimensionality of the embedding space (e.g., 2048 dimensions), embeddings from different tokens may be highly uncorrelated. Simply averaging these embeddings can distort the original distribution. For instance, consider two uncorrelated distributions $A \sim \mathcal{N}(\mu, \sigma^2)$ and $B \sim \mathcal{N}(\mu, \sigma^2)$; mean pooling would alter the original distribution to $\frac{A+B}{2} \sim \mathcal{N}(\mu, \frac{\sigma^2}{2})$, effectively scaling the variance by $\frac{1}{2}$. We found that, for most pre-trained LLMs, the distributions of embeddings are centered at $\mu \approx 0$. Thus, to prevent distortion of the original embedding distribution of LLMs, the Embedding Compress module only scales the sum of the c embeddings by $\frac{1}{\sqrt{c}}$.

Intuitively, it could be difficult for LLMs to understand these compressed embeddings. A simple approach is to supervise $\mathcal{M}$ to predict answers with a language modeling loss, which enforces $\mathcal{M}$ to model answers with compressed embedding inputs. However, this objective provides supervision signals that are too sparse to converge to near-optimal performance. To address this issue, we train

CoLaR to predict the compressed reasoning tokens. Ideally, when using a compression factor c , CoLaR should be able to *read* and *predict* tokens in groups of c . This means that for each compressed embedding input, CoLaR should be trained to predict all c corresponding tokens. To approximate this multi-label classification task using the single-label prediction capability of LLM’s language model head, we randomly sample one token from each group of c reasoning tokens $t_r^{k \times c: (k+1) \times c}$ as the ground-truth label. This approach trains the predicted logits to approximate a multimodal distribution that represents all potential tokens in each compressed group. This process could be formally represented as:

$$\mathcal{L}_{\text{comp}} = -\frac{1}{L_a + L_c} \sum_{i=1}^{L_a + L_c} \log p([\mathbf{t}_c, \mathbf{t}_a]^i | [\mathbf{e}_c, \mathbf{e}_a]^{1:i-1}, \mathbf{e}_q), \quad (1)$$

where $\mathbf{t}_c$ are sampled from $\mathbf{t}_r$.

3.2 Next compressed embedding prediction

To enable auto-regressive latent reasoning, we train a Latent Head $\mathcal{E}$ (analogous to the Language Head in LLMs) to predict the next compressed embedding, where $\mathcal{E}$ is a two-headed MLP. Given the current hidden states h_c^i output by $\mathcal{M}$, the Latent Head $\mathcal{E}$ predicts both the mean μ_c^{i+1} and standard deviation σ_c^{i+1} of the next embedding’s distribution.

Unlike previous works that predict deterministic values—which limits exploration of alternative reasoning pathways—our approach generates a probabilistic distribution. During inference, we employ the re-parameterization trick to sample the next embedding: $\hat{e}_c^{i+1} = \hat{\mu}_c^{i+1} + \hat{\sigma}_c^{i+1}\epsilon$, where ϵ is random noise sampled from a standard Gaussian distribution $\mathcal{N}(0, 1)$.

The Latent Head $\mathcal{E}$ is primarily trained using the negative log-likelihood (NLL) loss. For a prediction at position i , this can be formulated as:

$$\mathcal{L}_{\text{latent}}(i) = -\log p(e_c^i | \hat{\mu}_c^i, \hat{\sigma}_c^i) = \frac{(e_c^i - \hat{\mu}_c^i)^2}{2\hat{\sigma}_c^i} + \log \hat{\sigma}_c^i \quad (2)$$

This probabilistic formulation enables the model to capture uncertainty in the latent reasoning process and allows for diverse reasoning pathways during generation. The total loss is computed by averaging over all positions in the compressed embedding sequence.

However, we empirically found that CoLaR with NLL loss tends to under-fit on *simpler* math reasoning datasets that require less exploration. To address this, we propose a soft-MSE loss that combines Mean Squared Error with an entropy regularization term:

$$\mathcal{L}_{\text{latent}}(i) = \underbrace{\mathbb{E}_{\epsilon} [(\hat{\mu}_c^i + \hat{\sigma}_c^i \epsilon - e_c^i)^2]}_{\text{MSE term}} - \alpha \underbrace{\left(\frac{1}{2} \log(2\pi e (\hat{\sigma}_c^i)^2) \right)}_{\text{entropy term}}, \quad (3)$$

where α is a positive hyperparameter that encourages the model to predict more diverse latents with larger $\hat{\sigma}$ values. This approach enables CoLaR to better fit simpler datasets while maintaining its exploration capability. We evaluate **both the two forms** of latent loss in our experiments. We sum up $\mathcal{L}_{\text{comp}}$ and $\mathcal{L}_{\text{latent}}$ as the final loss to optimize CoLaR in the SFT stage.

3.3 Exploration with reinforcement learning

The next-compressed embedding prediction training enables latent reasoning chains to *mimic* original chain of thoughts. However, this paradigm inevitably limits the performance of latent reasoning models to their CoT teachers. To further explore the potential of CoLaR, we apply a reinforcement learning stage to our proposed method.

With the Latent Head trained to predict distributions, we can sample diverse latent reasoning pathways and final answers for the same question q . We then apply Group Relative Policy Optimization (GRPO) algorithm [23] to reinforce correct reasoning chains and answers while penalizing incorrect ones.

Specifically, for each question q , GRPO first samples a group of outputs $\{o_1, o_2, \dots, o_G\}$ from the old policy $\pi_{\theta_{\text{old}}}$, where G is the group size. Each output o_i consists of a latent reasoning chain and a final answer. Then, GRPO optimizes the policy π_{θ} by minimizing the following objective:

$$\mathcal{L}_{\text{GRPO}} = -\frac{1}{G} \sum_{i=1}^G \left(\min \left(\frac{\pi_{\theta}(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)} A_i, \text{clip} \left(\frac{\pi_{\theta}(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1 - \epsilon, 1 + \epsilon \right) A_i \right) \right), \quad (4)$$

where ϵ is a hyperparameter, and A_i is calculated as a group-normalized reward:

$$A_i = \frac{r_i - \text{mean}(r_1, r_2, \dots, r_G)}{\text{std}(r_1, r_2, \dots, r_G)}. \quad (5)$$

We simply set r_i to 1 when an answer is correctly predicted and to 0 otherwise. Following DAPO [37], we remove the KL-regularization term from original GRPO implementation for efficient training.

Notably, $\mathcal{L}_{\text{GRPO}}$ is calculated at the output level, i.e., across the entire latent reasoning chain and predicted answer, but is then **averaged** when applied to each latent/token. This design encourages CoLaR to balance exploration and exploitation. For instance, in Figure 2, although both $a_1 = a_2 = -1$, GRPO penalizes the latents/tokens in o_1 more, as there are fewer reasoning steps. This encourages CoLaR to think more deeply to **explore** correct reasoning paths. Likewise, the latents/tokens in o_4 are reinforced more as the reward is averaged less, which encourages CoLaR to **exploit** those more compact latent reasoning paths.

4 Experiments

In this section, we evaluate our proposed method CoLaR against strong baselines, analyze the contributions of different components, and explore the impact of key parameters.

4.1 Experimental setup

Datasets and tasks. Our method is mainly trained and evaluated on **GSM8k-Aug** [6], an augmented version of the Grade-School level Math reasoning dataset GSM8k [4]. GSM8k-Aug comprises approximately 385k training samples and 1k test samples. We also evaluate the trained methods on three out-of-domain math reasoning datasets: (1) **GSM-Hard** [9], a modified version of GSM8K with approximately 1k test samples featuring larger magnitude numbers, (2) **SVAMP** [20] and (3) **MultiArith** [21], two simpler math reasoning datasets with 1k and 600 test samples, respectively. Moreover, we train and evaluate our method on a more challenging dataset **MATH** [14], which consists of 7.5k training samples and 5k test samples, covering algebra, calculus, statistics, geometry, linear algebra and number theory. Following [13], we use two metrics: (1) Accuracy (Acc.), which measures the effectiveness of correctly predicting answers and (2) Reasoning chain length (# L), which measures efficiency by averaging the number of tokens/latents predicted in reasoning chains.

Baseline methods. We primarily compare against the following baselines: (1) **CoT** [29], which is fine-tuned on complete reasoning chains and answers, and performs token-level reasoning before predicting answers during inference; (2) **iCoT** [5], which internalizes reasoning knowledge by gradually removing reasoning steps, and directly predicts answers during inference; (3) **Coconut** [13], which is fine-tuned with a curriculum process to gradually replace token-level reasoning steps with latent reasoning steps, and performs six steps of latent reasoning before predicting answers; and (4) **Distill**, is our reproduced version of CODI [24] based on their implementation details as the code and model are not released. It self-distills token-level CoT into fixed-length latent reasoning steps, with an inference stage same to Coconut.

Implementation details. (1) Base model: unless otherwise specified, all experiments use a frozen Llama-3.2-1B-Instruct [11] backbone with a trainable LoRA module [15]. Following Coconut, all methods are initialized with weights from CoT-SFT to accelerate training. (2) Model checkpointing: for fair comparison, all models are trained for up to 50 epochs or 12 hours, whichever is reached first, and we choose the checkpoint that achieves the best accuracy on the validation set as the final model. (3) Hyper-parameters: we use the AdamW [18] optimizer with a fixed learning rate of 1e-4 and a weight decay of 1e-2 in SFT stage, and set the learning rate to 1e-6 in RL stage. We set $r_{\text{max}} = 5$ to train CoLaR. During inference, we configure the LLM generation with a temperature of 1 and top-p of 0.9. All SFT training processes are conducted a total batch size of 256. For more implementation details, please refer to Appendix Section A.

Table 1: Experiment results of baseline methods and CoLaR on four grade-school math reasoning datasets. We test the methods for five times with different random seeds to report the averaged number and 95% confidence interval ($\pm$) on accuracy (Acc. %) and reasoning chain length (# L). CoLaR- c denotes a same CoLaR model tested with different compression factors c . For ablation methods (marked in gray), suffixes DL, OC, MP and NLL denote CoLaR with a Deterministic Latent head, training with Out Compressed reasoning chain in cross entropy labels, using Mean Pooling to compress embeddings, and training with NLL loss, respectively.

	GSM8k-Aug		GSM-Hard		SVAMP		MultiArith		Average	
	Acc.	# L	Acc.	# L	Acc.	# L	Acc.	# L	Acc.	# L
CoT	49.4 $\pm$.72	25.6 $\pm$.11	11.9 $\pm$.16	34.2 $\pm$.11	59.8 $\pm$.29	12.1 $\pm$.03	93.2 $\pm$.49	13.7 $\pm$.09	53.6	21.4
iCoT	19.8 $\pm$.23	0.00 $\pm$.00	3.87 $\pm$.16	0.00 $\pm$.00	36.4 $\pm$.51	0.00 $\pm$.00	38.2 $\pm$.66	0.00 $\pm$.00	24.6	0.00
Coconut	23.1 $\pm$.28	6.00 $\pm$.00	5.49 $\pm$.33	6.00 $\pm$.00	40.7 $\pm$.65	6.00 $\pm$.00	41.1 $\pm$.24	6.00 $\pm$.00	27.6	6.00
Distill	13.3 $\pm$.62	6.00 $\pm$.00	2.97 $\pm$.24	6.00 $\pm$.00	21.7 $\pm$.73	6.00 $\pm$.00	19.2 $\pm$.83	6.00 $\pm$.00	14.3	6.00
CoLaR-5	26.8 $\pm$.17	5.57 $\pm$.02	5.87 $\pm$.10	6.53 $\pm$.01	48.4 $\pm$.45	2.95 $\pm$.02	86.4 $\pm$.35	3.21 $\pm$.01	41.7	4.57
- DL	26.7 $\pm$.11	5.74 $\pm$.01	5.53 $\pm$.11	8.20 $\pm$.04	48.3 $\pm$.05	2.90 $\pm$.01	84.5 $\pm$.19	3.22 $\pm$.01	41.3	5.02
- OC	24.8 $\pm$.27	5.14 $\pm$.12	6.46 $\pm$.11	5.49 $\pm$.06	46.5 $\pm$.18	2.85 $\pm$.01	85.9 $\pm$.22	3.13 $\pm$.01	40.1	4.15
- MP	20.6 $\pm$.22	5.61 $\pm$.02	4.20 $\pm$.07	6.18 $\pm$.02	47.7 $\pm$.41	2.96 $\pm$.01	80.7 $\pm$.59	3.20 $\pm$.01	38.3	4.49
- NLL	20.3 $\pm$.64	5.99 $\pm$.06	4.52 $\pm$.39	16.6 $\pm$.25	43.9 $\pm$.43	3.06 $\pm$.03	81.6 $\pm$.23	3.20 $\pm$.02	37.6	8.01
CoLaR-2	40.1 $\pm$.20	12.7 $\pm$.02	9.08 $\pm$.03	14.0 $\pm$.07	54.9 $\pm$.20	6.11 $\pm$.01	91.3 $\pm$.12	7.35 $\pm$.01	48.8	10.0
- DL	39.7 $\pm$.11	12.8 $\pm$.01	8.84 $\pm$.06	17.2 $\pm$.09	54.3 $\pm$.23	6.10 $\pm$.01	90.1 $\pm$.17	7.46 $\pm$.01	48.2	10.9
- OC	39.1 $\pm$.33	12.3 $\pm$.04	8.96 $\pm$.01	16.9 $\pm$.13	54.7 $\pm$.18	6.08 $\pm$.02	90.1 $\pm$.25	7.36 $\pm$.01	48.2	10.6
- MP	36.9 $\pm$.30	12.4 $\pm$.02	8.46 $\pm$.19	12.0 $\pm$.05	54.1 $\pm$.42	6.14 $\pm$.01	86.8 $\pm$.20	7.43 $\pm$.01	46.6	9.49
- NLL	32.3 $\pm$.51	12.2 $\pm$.04	7.57 $\pm$.16	16.6 $\pm$.25	51.0 $\pm$.24	5.50 $\pm$.03	88.3 $\pm$.41	7.09 $\pm$.02	44.8	10.3

4.2 Comparison to baseline methods on GSM datasets

Table 1 presents a comparison of CoLaR against state-of-the-art baseline methods on four grade-school level math reasoning datasets. CoLaR demonstrates consistent performance gains over existing latent-based reasoning approaches. Notably, CoLaR with a test-time compression factor of 5 (CoLaR-5) achieves a 14.1% improvement in average accuracy compared to Coconut, and does so with fewer reasoning steps (4.57 vs. 6.00). This advantage stems from our effective next-compressed embedding prediction objective, which efficiently compresses the reasoning process into compact and informative latent variables. This allows for superior performance while maintaining a high compression ratio.

Leveraging the dynamic compression design, we evaluated the same trained model with a different test-time compression factor of 2 (CoLaR-2). The resulting accuracy of 48.8% represents only a 4.8% decrease compared to explicit CoT, but with a significant 53% reduction in reasoning chain length.

Furthermore, CoLaR exhibits robust out-of-domain generalization capabilities compared to other latent-based baselines, particularly on the MultiArith dataset, where CoLaR shows minimal performance degradation compared to CoT, while other latent-based methods suffer significant drops.

4.3 Ablation studies of on GSM datasets

We conduct ablation studies with four experimental settings on the GSM datasets, with results illustrated in the gray areas of Table 1. Three key findings emerge:

(1) **Simple math questions require balanced exploration capability.** Comparing CoLaR with CoLaR-DL (trained with a deterministic latent head) and CoLaR-NLL (trained with NLL loss), we find that a deterministic latent head fits well on simple math datasets but lacks test-time exploration potential, leading to suboptimal performance. Conversely, training CoLaR with NLL loss introduces excessive randomness, resulting in poor fit on the training data and worse overall performance.

(2) **Dense reasoning supervision signals are crucial.** When comparing CoLaR with CoLaR-OC, where we remove the tokens of compressed reasoning chain t_c and use only the final answer tokens t_a as the language modeling supervision signal, performance degrades by 1.6% and 0.6% at compression factors $c = 5$ and $c = 2$, respectively. This confirms the importance of dense supervision signals when training latent-based reasoning methods.

(3) **Latents under different compression factors should share the same space.** CoLaR-MP, which applies Mean Pooling on the compression embeddings, shows 3.4% and 2.2% performance degradation compared to our method. This decline is primarily attributed to distribution shifts caused by the compression process, which introduces confusion during model training.

Table 2: Experimental results on the challenging MATH dataset. We evaluate our proposed method CoLaR on two base models and three settings: -DL denotes using a Deterministic Latent head, -NLL denotes CoLaR trained with NLL Loss as $\mathcal{L}_{\text{latent}}$, which is our main method, and -/w GRPO denotes the post-trained CoLaR-NLL with GRPO reinforcement learning process. We calculate the performance gain between CoLaR-NLL and CoLaR-NLL-RL to highlight the effectiveness of reinforcement learning. Compression factor c and $\#L_{max}$ are set to 2 and 128, respectively.

	DeepSeek-R1-Distill-Qwen-1.5B		Llama-3.2-1B-Instruct	
	Acc.	#L	Acc.	#L
CoT	23.5 $\pm$.29	209 $\pm$ 1.6	9.71 $\pm$.33	210 $\pm$ 1.4
CoLaR-DL	9.04 $\pm$.12	99.4 $\pm$.25	3.07 $\pm$.28	134 $\pm$.46
CoLaR-NLL	8.94 $\pm$.21	56.8 $\pm$.14	5.28 $\pm$.16	83.1 $\pm$.52
CoLaR-NLL-RL	14.3 $\pm$.25 (5.36% $\uparrow$)	9.79 $\pm$.40 (82.8% $\downarrow$)	7.08 $\pm$.07 (1.80% $\uparrow$)	16.1 $\pm$.14 (80.6% $\downarrow$)
- w/o average	13.8 $\pm$.14	128 $\pm$.00	0.00 $\pm$.00	128.0 $\pm$.00

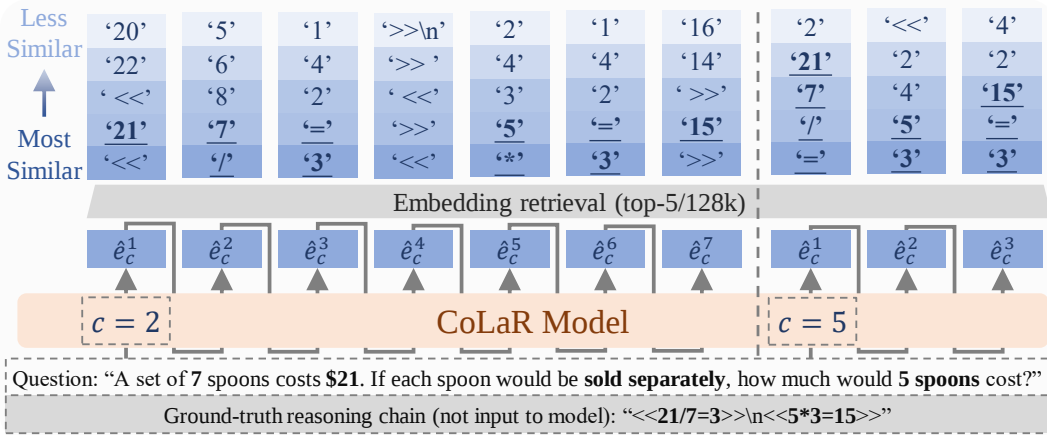


Figure 3: A case study on the GSM-8k validation set. We set the compression factor c to 2 and 5, which produce two latent reasoning chains in length 7 and 3, respectively. We then retrieve tokens with the predicted latents by embedding cosine similarity, and underscore those informative tokens.

4.4 Reinforcement learning results on the MATH dataset

We train and evaluate CoLaR with RL on the challenging MATH dataset, using two base models [12, 34, 11]. The results are presented in Table 2. Our analysis of the results yields three key conclusions:

(1) **Exploration is crucial for difficult problems.** The deterministic latent reasoning process of CoLaR-DL exhibits accuracy comparable to or worse than CoLaR-NLL despite longer reasoning chains. This suggests that challenging math problems necessitate exploration of multiple potential solutions, rather than deterministic, step-by-step reasoning. Furthermore, post-training CoLaR-NLL with RL yields significant gains, achieving up to 5.36% higher accuracy and an 82.8% reduction in reasoning length. This highlights the potential of RL and the importance of balancing exploration and exploitation for latent reasoning models.

(2) **Averaged rewards promote exploitation.** When training without this averaging (i.e., simply dividing the loss by a constant to normalize the loss scale), we observed that while Qwen-1.5B exhibited a performance increase (from 8.94% to 13.8%) similar to averaging the loss, the reasoning length rapidly converged to the pre-defined upper limit. Moreover, Llama-1B’s performance tend towards collapse. This suggests the averaged design encourages CoLaR to exploit more efficient reasoning pathways.

(3) **Base model quality impacts RL effectiveness.** Supervised fine-tuning on CoT resulted in varying performance across the two base LLMs. Meanwhile, CoLaR also demonstrates a significantly larger performance gain on RL when using the higher-quality Qwen-1.5B compared to Llama-1B. This

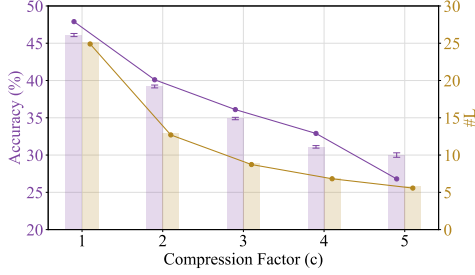


Figure 4: Accuracy and reasoning chain length (# L) of CoLaR on GSM8k dataset when trained with random $c \in [1, 5]$ (the **lines**) or trained solely on specific c (the **bars**).

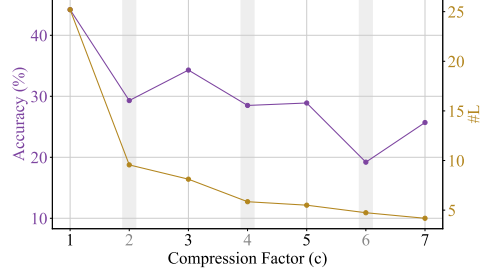


Figure 5: Accuracy and reasoning chain length (# L) of CoLaR on GSM8k dataset when trained with $c \in \{1, 3, 5, 7\}$ and tested with extra $c \in \{2, 4, 6\}$ (under gray bars).

Table 3: Experimental results of CoLaR-2 and TokenSkip on GSM-8k, MATH-500, and GPQA.

Acc. (%)#L	CoT-1B	CoLaR-1B	CoT-8B	CoLaR-8B	CoLaR-8B-RL	TokenSkip-8B
GSM-8k	47.5/92.6	40.4/49.2	76.5/93.6	70.8/47.3	71.9/13.3	78.2/113
MATH-500	28.6/176	19.8/84.6	54.6/168	45.8/67.7	52.4/17.6	40.2/292
GPQA	26.4/232	26.4/84.1	35.7/216	32.4/101	37.5/66.7	-

observation aligns with the findings of [8], which suggests that RL substantially **activates inherent reasoning capabilities**, indicating the importance of base model quality.

We also observed that during the RL training process, CoLaR tends to *think longer* initially with a rapid rise in accuracy, followed by a phase of *thinking shorter* accompanied by a more stable increase in accuracy, aligning with the discussion in Section 3.3. Due to space constraints, the detailed training curves are provided in Appendix C.

4.5 Scaling CoLaR on larger base model

We scale CoLaR to the 1-Billion and 8-Billion parameter Llama checkpoints to verify that its compression and accuracy gains grow with model capacity. The 8B model also matches the backbone used by TokenSkip, allowing a direct comparison under its published protocol. Experiments are run on GSM-8k, MATH-500 which are slightly different to Section 4.1 to align with TokenSkip’s settings) and the out-of-domain GPQA benchmark (chemistry, biology and physics). The results are shown in Table 3. Moving from 1 B to 8 B parameters raises CoLaR’s accuracy on every dataset while keeping the reasoning chain roughly half the length of the CoT teacher. Reinforcement learning delivers further improvements: +1.1% on GSM-8k, +6.6% on MATH-500 and +5.1% on GPQA, again with large additional compression. Most notably, CoLaR-8B-RL reaches 37.5% on GPQA, surpassing its 35.7% CoT teacher and shortening the reasoning chain by 69%, confirming that latent RL remains effective as models grow.

4.6 Interpreting chains of latent thoughts

To quantitatively illustrate the effectiveness of our proposed CoLaR, and make the latent reasoning process more transparent, we conduct a case study of CoLaR on the GSM8k validation set. The results are illustrated in Figure 3.

As latent variables are fundamentally scaled sum of word token embeddings, we could directly calculate the cosine similarity between latent variables and the entire LLM embedding matrix:

- When prompted with compression factor $c = 2$, CoLaR auto-regressively produces seven latent variables, and then automatically terminates reasoning process. By calculating similarity scores, each latent variable is capable of retrieving meaningful words such as “«”,

“2/”, and the entire chain of latent thoughts could be interpreted as “ $\langle 2/7=3 \rangle \langle 5*3=15 \rangle$ ”, which exactly matches the correct calculation process.

- When prompted with higher compression factor $c = 5$, less informative tokens such as “ $\langle$ ” are ignored, demonstrating both the effectiveness and efficiency of our proposed dynamic compression mechanism.

4.7 Analyses on dynamic compression factors

We investigate the generalization capability of CoLaR across different compression factors c . Two key findings emerge from our analyses:

First, as illustrated in Figure 4, for each test-time compression factor (except from $c = 5$), CoLaR trained with random $c \in [1, 5]$ consistently outperforms models trained on a single compression factor. These results demonstrate that exposure to diverse training-time compression factors produces complementary benefits for generalization. For example, training with $c = 2$ also improves the performance of testing with $c = 4$, highlighting the effectiveness of our dynamic training process.

Second, as shown in Figure 5, we train CoLaR with $c \in \{1, 3, 5, 7\}$ and evaluate it with previously unseen compression factors $c \in \{2, 4, 6\}$. We find that CoLaR successfully generalizes to these unseen compression factors, maintaining expected actual compression rates. Moreover, though worse in absolute values, the slope of the performance curve on out-of-domain compression factors closely resembles that of in-domain factors, suggesting robust interpolation capabilities.

5 Limitations

While CoLaR demonstrates superior effectiveness and efficiency in latent reasoning, we acknowledge two important limitations: (1) Though CoLaR surpassing its CoT teacher on GPQA, on the remaining benchmarks, the overall performance of CoLaR currently *approximates* explicit CoT reasoning without surpassing it. (2) We observe that CoLaR struggles to generalize to non-integer compression factors (e.g., $c = 1.5$) or to values greater than the maximum training compression factor r_{max} . This limitation stems primarily from the discrete tokenization constraints inherent to large language models, which restrict the continuous representation of compression factors. (3) Beyond technical limitations, our work on enhancing reasoning capabilities in LLMs has significant societal implications. On the positive side, CoLaR could significantly boost the efficiency of existing LLM services. However, potential negative impacts include the risk of amplifying existing biases in reasoning processes and possible misuse for generating more convincing misinformation. To mitigate these risks, we recommend careful monitoring of downstream applications.

6 Conclusion

In this paper, we introduce Compressed Latent Reasoning (CoLaR), a framework that dynamically compresses LLM reasoning chains into latent space while maintaining exploration-exploitation capabilities. Our method centers on three key innovations: (1) compressed latent reasoning through an auxiliary next compressed embedding prediction task that encapsulates the semantics of multiple tokens, (2) dynamic training and inference with variable compression factors that allows for flexible reasoning chain lengths and fully parallelized processing, and (3) a probabilistic latent head for reinforcement learning that enables exploration of diverse reasoning pathways for higher accuracy while exploiting shorter reasoning chains for efficiency. Our experimental results demonstrate that CoLaR achieves a 14.1% improvement in accuracy compared to state-of-the-art latent-based reasoning methods, while reducing reasoning chain length by 53.3% with only a 4.8% performance degradation relative to explicit CoT. On the challenging MATH dataset, reinforcement learning techniques further boost CoLaR’s performance by 5.36% while dramatically reducing reasoning chain length by 82.8%. Future work will focus on addressing non-integer compression factors, exploring more sophisticated reinforcement learning approaches, and extending our dynamic compression mechanism to more diverse reasoning tasks beyond mathematics.

Acknowledgements. This work is supported by the Beijing Natural Science Foundation (L233008) and Xiaomi Inc. We acknowledge the anonymous reviewers for their helpful comments.

References

- [1] Simon A Aytes, Jinheon Baek, and Sung Ju Hwang. Sketch-of-thought: Efficient llm reasoning with adaptive cognitive-inspired sketching. *arXiv preprint arXiv:2503.05179*, 2025.
- [2] Yilong Chen, Junyuan Shang, Zhenyu Zhang, Yanxi Xie, Jiawei Sheng, Tingwen Liu, Shuohuan Wang, Yu Sun, Hua Wu, and Haifeng Wang. Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. *arXiv preprint arXiv:2502.13842*, 2025.
- [3] Jeffrey Cheng and Benjamin Van Durme. Compressed chain of thought: Efficient reasoning through dense representations. *arXiv preprint arXiv:2412.13171*, 2024.
- [4] 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.
- [5] Yuntian Deng, Yejin Choi, and Stuart Shieber. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*, 2024.
- [6] 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.
- [7] Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Efficient reasoning models: A survey. *arXiv preprint arXiv:2504.10903*, 2025.
- [8] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *arXiv preprint arXiv:2503.01307*, 2025.
- [9] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.
- [10] Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. *arXiv preprint arXiv:2310.02226*, 2023.
- [11] 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.
- [12] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- [13] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.
- [14] 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. *arXiv preprint arXiv:2103.03874*, 2021.
- [15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.
- [16] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- [17] Zongzhao Li, Zongyang Ma, Mingze Li, Songyou Li, Yu Rong, Tingyang Xu, Ziqi Zhang, Deli Zhao, and Wenbing Huang. Star-r1: Spatial transformation reasoning by reinforcing multimodal llms. *arXiv preprint arXiv:2505.15804*, 2025.

- [18] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [19] Amirkeivan Mohtashami, Matteo Pagliardini, and Martin Jaggi. Cotformer: More tokens with attention make up for less depth. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ NeurIPS 2023)*, 2023.
- [20] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191*, 2021.
- [21] 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, 2015.
- [22] Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*, 2025.
- [23] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- [24] Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. *arXiv preprint arXiv:2502.21074*, 2025.
- [25] DiJia Su, Hanlin Zhu, Yingchen Xu, Jiantao Jiao, Yuandong Tian, and Qinqing Zheng. Token assorted: Mixing latent and text tokens for improved language model reasoning. *arXiv preprint arXiv:2502.03275*, 2025.
- [26] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.
- [27] Wenhui Tan, Boyuan Li, Chuhao Jin, Wenbing Huang, Xiting Wang, and Ruihua Song. Think then react: Towards unconstrained action-to-reaction motion generation. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [28] Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1.5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.
- [29] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [30] Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*, 2025.
- [31] Violet Xiang, Charlie Snell, Kanishk Gandhi, Alon Albalak, Anikait Singh, Chase Blagden, Duy Phung, Rafael Rafailov, Nathan Lile, Dakota Mahan, et al. Towards system 2 reasoning in llms: Learning how to think with meta chain-of-thought. *arXiv preprint arXiv:2501.04682*, 2025.
- [32] Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.
- [33] Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot: Soft chain-of-thought for efficient reasoning with llms. *arXiv preprint arXiv:2502.12134*, 2025.
- [34] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

- [35] Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Zheng Lin, Li Cao, and Weiping Wang. Dynamic early exit in reasoning models. *arXiv preprint arXiv:2504.15895*, 2025.
- [36] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.
- [37] Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.
- [38] Yufeng Yuan, Qiyang Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, Tiantian Fan, Zhengyin Du, Xiangpeng Wei, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.
- [39] Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. Quiet-star: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and introduction clearly articulate our main contribution of developing a novel approach for compressing explicit reasoning chains into latent space, accurately reflecting the scope of theoretical foundations and empirical results across the mathematical reasoning benchmarks evaluated in our experiments.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: There is a separate "Limitations" section discussion the main limitations on performance compared to explicit CoT methods, and worse generalization capability on certain compression factors.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This paper presents a framework for latent reasoning compression but does not include formal theorems with proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Our paper provides comprehensive implementation details necessary for reproducing the proposed method. We include detailed descriptions of the model architecture, training procedure, hyperparameters, and evaluation methodology. Additionally, we will open source our complete codebase, including the environment setup, model implementation, training pipeline, and evaluation framework, to facilitate direct replication of our experimental results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

- (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We provide comprehensive implementation details to reproduce our method, but we have chosen not to include anonymous URLs to our code repository at the submission stage, to balance scientific transparency with the appropriate protection of intellectual property. We make a strong commitment to release our complete codebase immediately upon acceptance of the paper, including model implementation, training procedures, evaluation framework, and necessary scripts to reproduce all experimental results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide comprehensive methodological details, hyperparameter specifications, and experimental protocols for implementation in the manuscript and appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All experimental results in our paper are reported as averages across five independent runs with different random seeds. We include 95% confidence intervals for all numerical results, calculated using the standard error of the mean multiplied by 1.96. These confidence intervals are visually represented in our figures as error bars and explicitly stated with $\pm$ in our tables.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide complete information about the computational resources used in our experiments in the Appendix, including hardware setup, virtual environment configuration, and total training hours.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: Our research fully complies with the NeurIPS Code of Ethics through our transparent methodology, proper acknowledgment of prior work, and responsible data usage. We have documented our approach thoroughly, and maintained scientific integrity throughout our work on enhancing reasoning capabilities in Large Language Models.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[Yes\]](#)

Justification: Our paper discusses both positive societal impacts (improved reasoning capabilities that could enhance decision support systems and educational tools) and potential negative implications (risks of amplifying existing biases in reasoning processes and possible misuse for generating misleading content). We outline mitigation strategies including careful monitoring of downstream applications.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [\[NA\]](#)

Justification: Our paper is primarily analytical in nature, focusing on methods to enhance and evaluate reasoning capabilities within existing models rather than releasing new models.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [\[Yes\]](#)

Justification: The pre-trained LLMs and datasets we use are fully open-sourced and correctly cited in this paper.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[NA\]](#)

Justification: We conduct experiments using existing pre-trained LLMs and datasets, but do not create new assets as part of this publication.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [\[NA\]](#)

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Since LLMs form the core subject of our research, we have thoroughly documented all aspects of their usage in our methodology section.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

A More implementation details

In this section, we provide comprehensive details regarding our model architecture, training hyperparameters, and dataset specifications.

Model hyperparameters. For our experiments, we employ either a frozen Llama-3.2-1B-Instruct or DeepSeek-distill-Qwen-1.5B as our LLM backbone, augmented with a tunable LoRA module. All LoRA modules are configured with $\alpha = 32$ and $r = 128$ consistently across all experiments. Our method incorporates a Latent Head, implemented as a three-layered MLP with hidden dimensions corresponding to the LLM backbone’s dimension ($d = 2048$).

Training hyperparameters. We utilize the AdamW optimizer with a weight decay of $1e-2$ throughout our experiments. The learning rate is set at $1e-4$ for supervised fine-tuning (SFT) and $1e-6$ for reinforcement learning (RL). For SFT experiments, we leverage Distributed Data Parallel across eight A100 GPUs with a total batch size of 256. The RL experiments are conducted on a single A100 GPU with a rollout batch size of 8, optimizer step batch size of 4, group size G of 8, and clip ϵ of 0.2. To ensure reproducibility, we fix the random seed of all libraries (Python, CUDA, PyTorch, and NumPy) to 0 for training processes. For evaluation, we use five distinct runs with random seeds sequentially set from 0 to 4.

Notably, when training the Latent Head, we normalize the target (i.e., the ground-truth compressed embeddings) to ensure training stability. This normalization is implemented by dividing the target by the standard deviation σ_e of the embeddings. Since the embedding distributions are already centered at approximately zero ($\mu \approx 0$), we do not apply any shift during normalization. During inference, we multiply the predicted embeddings by the standard deviation to rescale them to match the LLM’s original embedding distribution. These statistics can be either learned during training or calculated in advance; we opt for the latter approach for simplicity. We observe model-specific values, with $\sigma_e \approx 0.02$ for Llama-3.2-1B-Instruct and $\sigma_e \approx 0.03$ for Qwen-1.5B. This normalization process is critical for maintaining numerical stability while preserving the relative relationships between embedding dimensions.

Dataset information. We evaluate our method on four grade-school mathematics datasets—GSM8K-Aug, GSM8k-Hard, SVAMP, and MultiArith—as well as the more challenging MATH dataset for advanced mathematical reasoning. Since the original MATH dataset does not provide an official validation set, we randomly shuffle the training set and allocate 10% of the samples for validation purposes.

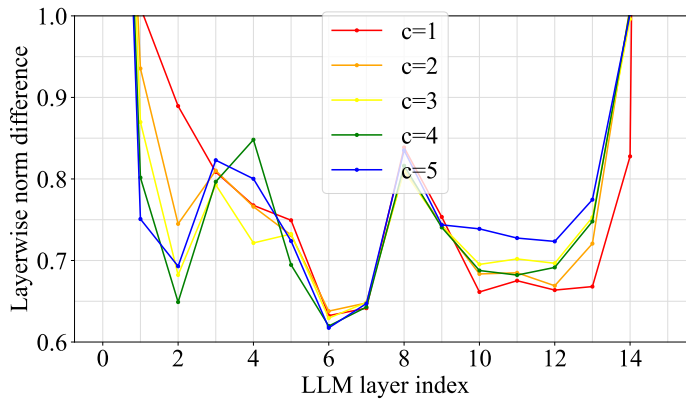


Figure 6: Layer-wise norm differences from CoLaR-1 to CoLaR-5.

B Layer-wise analyses on compression factors

We further investigate how the compression factor c influences activation patterns across LLM layers, with results shown in Figure 6. Specifically, we tested CoLaR on the same sample as in Section 4.6 with compression factors ranging from 1 to 5, calculating the relative activation norm differences between consecutive LLM layers.

Our analysis reveals distinct patterns across different network depths:

- **Shallow layers** (0-3, near input): CoLaR shows higher **activation** on smaller compression factors with more pronounced layer-wise changes in magnitude.
- **Intermediate layers** (3-9): Models with different compression factors exhibit similar behavior.
- **Deeper layers** (9-15, near output): Higher compression factors maintain stronger activation patterns.

This phenomenon can be explained as follows: when predicting less informative tokens (e.g., “<”) with lower compression factors (especially with $c = 1$, which uses no compression), the model requires minimal “thinking” and can determine the next token using primarily shallow layers. Consequently, computation in deeper layers is largely *underutilized*.

In contrast, higher compression factors enable CoLaR to process information more **densely**, with each latent representation carrying richer semantic content. This requires deeper layers to remain actively engaged in analyzing the condensed information and predicting subsequent compressed latents, thereby making more efficient use of the model’s computational capacity. These findings align with observations from previous work on internal thinking processes in transformer models [2, 22].

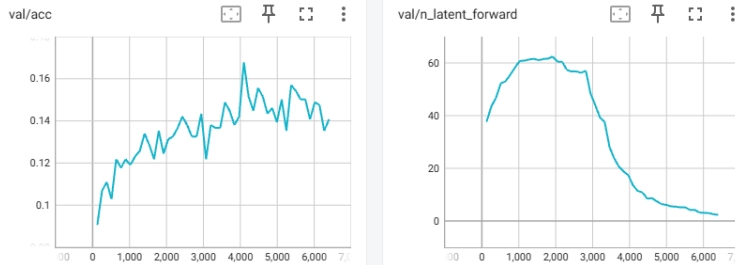


Figure 7: The validation accuracy and latent reasoning chain length curve on MATH dataset.

C RL training curves

Figure 7 presents the training curves from our reinforcement learning phase. The accuracy on the validation set exhibits a distinct three-phase pattern. In the initial exploration phase, accuracy increases rapidly from 9% to 14%, accompanied by an expansion of latent reasoning steps from 40 to 60. During this phase, the GRPO algorithm primarily encourages CoLaR to explore more extensively to discover correct reasoning pathways.

In the subsequent exploitation phase, validation accuracy fluctuates between 14% and 16%, while the latent reasoning length decreases from 60 to 20. With the per-token averaged reward/loss, the GRPO algorithm reinforces CoLaR to exploit shorter yet effective reasoning pathways.

Finally, as CoLaR begins to overfit, our early-stopping strategy is triggered to preserve the best-performing checkpoint at approximately 4k steps.

D Scaling properties of CoLaR

Figure 8 illustrates the performance characteristics of CoLaR when implemented with foundation models of varying parameter counts, ranging from 1 billion to 8 billion parameters. Our results

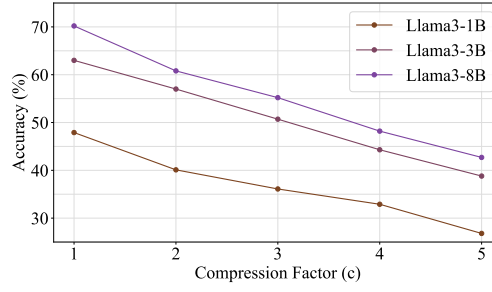


Figure 8: Performance of CoLaR when implemented with base LLMs ranging from 1B to 8B parameters.

demonstrate that CoLaR follows established neural scaling laws, with performance improvements correlating predictably with increases in the underlying model size. This consistent scaling behavior suggests that the benefits of our approach extend proportionally across different model scales, indicating CoLaR’s architectural effectiveness is not limited to specific parameter regimes.