

---

# Activation Control for Efficiently Eliciting Long Chain-of-thought Ability of Language Models

---

Zekai Zhao\*, Qi Liu\*, Kun Zhou<sup>†</sup>, Zihan Liu, Yifei Shao, Zhiting Hu, Biwei Huang  
University of California, San Diego.  
kuzhou@ucsd.edu

## Abstract

Despite the remarkable reasoning performance, eliciting the long chain-of-thought (CoT) ability in large language models (LLMs) typically requires costly reinforcement learning or supervised fine-tuning on high-quality distilled data. We investigate the internal mechanisms behind this capability and show that a small set of high-impact activations in the last few layers, greatly govern the long-form reasoning attributes, *e.g.*, output length and self-reflection. Through simply amplifying these activations and adding “wait” tokens, the long CoT ability can be invoked without training, leading to significantly increased self-reflection rate and accuracy. In addition, we also find that the activation changes follow predictable trajectories, *i.e.*, a sharp rise after special tokens and a subsequent exponential decay. Based on these insights, we introduce a general training-free activation control technique. It utilizes a few contrastive examples to identify the relevant activations, and then incorporates simple analytic functions to adjust their values at inference time to elicit long CoTs. Extensive experiments have verified the effectiveness of our methods in efficiently eliciting the long CoT ability of LLMs and improving the performance. Besides, we further propose a parameter-efficient fine-tuning method that trains only the last-layer activation amplification module and a few LoRA layers, outperforming LoRA on reasoning benchmarks with much fewer parameters. Our code and data are fully public released at <https://github.com/ZekaiZ123/EELo-CoT>.

## 1 Introduction

On the path to artificial general intelligence (AGI), enhancing the reasoning ability of large language models (LLMs) [1, 2, 3, 4] remains to be one of the most important challenges. Techniques like chain-of-thought (CoT) prompting [5] can elicit good performance on reasoning tasks by prompting the LLM to generate intermediate reasoning steps. However, in complex reasoning tasks (*e.g.*, math competition problems), such a way is prone to make mistakes in intermediate steps, finally failing to reach the accurate answer. Recently, long-CoT models and systems such as OpenAI-o1 [3] and DeepSeek-R1 [6] have exhibited remarkable performance in solving complex tasks. These methods can perform human-like slow-thinking behaviors, with quite a few careful deliberation and self-reflection steps before generating the final answer [6].

To elicit the long CoT ability, existing work relies on either reinforcement learning (RL) on high-quality instances with proper annotations, or supervised fine-tuning (SFT) on distilled data [7, 8, 9]. However, in practice, it is costly to collect enough well-annotated high-quality instances or distilled data [8], especially for complex reasoning tasks that humans may also not perform well. Besides, the RL process is also complicated, and requires expert knowledge to control the training stability and

---

\*Equal Contribution

<sup>†</sup>Corresponding Author

effectiveness [10, 11]. In fact, the key bottleneck comes from the lack of understanding of *how the long CoT ability is elicited in LLMs*. Since LLMs have undergone a very long pre-training stage and then a relatively short post-training stage using RL or SFT, it is possible that the long-CoT ability already exists in base models [6, 12]. If so, we can efficiently wake up the long-CoT ability when necessary, and exert influence to achieve more fine-grained controls in base models.

In LLM, activation values of MLP layers play a key role in preserving key information or styles during inference [13, 14]. Inspired by it, we analyze and compare the activation dynamics in state-of-the-art LLMs with and without the long-CoT ability during inference on complex reasoning tasks. By tracing the layer-wise activations, we identify specific positions and patterns that correlate with desirable CoT traits (*e.g.*, longer and with self-reflection). These activations generally locate in the last few layers (see Fig. 1). By simply amplifying them and adding the “wait” token, the long CoT ability can be invoked without training (see Fig. 3). In addition, we also find that the activation changes in these positions follow predictable patterns, *i.e.*, a sharp rise after the trigger tokens (*e.g.*, wait) and a subsequent exponential decay (see Fig. 4). Therefore, it is promising to utilize an analytic function to fit the pattern and mimic the activation to efficiently elicit the long-CoT ability. Based on the above findings, we propose a training-free activation control approach, to elicit the long-CoT ability in the inference process. Concretely, we first collect a few pairs of contrast examples about the long-CoT ability. Then, we use the contrast examples to identify the related activation positions and collect patterns in the base model. Next, we formulate the pattern of each activation value into a unified function with few coefficients to control the change tendency and intervention intensity. Based on the collected patterns, we fit these coefficients, and devise a simple rule that triggers intervention after encountering special tokens during inference.

Since we do not need to train the LLM and the activation control method only requires using base models, our approach is general to any LLMs and any datasets for efficiently elicit the long-CoT ability when necessary. Extensive experiments conducted on complex math and science reasoning benchmarks have demonstrated the effectiveness of our approach in improving the LLM performance (see Table 3) with special long-CoT style. We also reveal that the long-CoT related activation positions become more inactive or even dead in over-posttrained LLMs.

Furthermore, as we can identify the key activations within the LLM, we also propose a parameter-efficient fine-tuning method that focuses on automatically learning the activation pattern from long-CoT related training data. Concretely, we add a rather lightweight learnable amplification module for the identified activations in the last layer, and add LoRA layers with a low rank in the former layers, while fixing the parameters of other parameters. In this way, our approach only requires training much fewer parameters than the existing LoRA method. Experimental results have shown that our approach can achieve better performance than LoRA. It achieves comparable performance of full-parameter fine-tuning method, but only trains 1.51% of total parameters.

## 2 Empirical Analysis on the Activation Patterns of Long CoT Ability

In this section, we empirically analyze the activation patterns underlying the long CoT reasoning ability, since existing work has shown that activation values can preserve key information of styles or concepts [13]. We conduct a series of experiments on state-of-the-art Qwen2.5-7B series LLMs, and first study (1) the distribution of long-CoT related activations, then test (2) whether activation intervention can elicit long-CoT ability, finally analyze (3) whether there are predictable dynamic patterns in activation values during inference.

### 2.1 Long-CoT Related Activations Distribution Study

We aim to reveal how the key activation values are distributed within an LLM and the difference between LLMs with and without the long-CoT ability. We first collect a set of contrastive example pairs about the long-CoT characteristics, which are fed to the LLM to collect the long-CoT related activations. Then, we analyze the distribution of the top ones across layers and different LLMs.

**Analysis Setup.** We conduct the experiments on MATH [15], a widely used mathematical reasoning dataset. For the contrastive example pairs, we consider the following three key characteristics of long-CoT ability, *i.e.*, longer, with self-reflection, and more accurate. Thus, we build two datasets that contain positive and negative samples that satisfy and unsatisfy the above features, respectively.

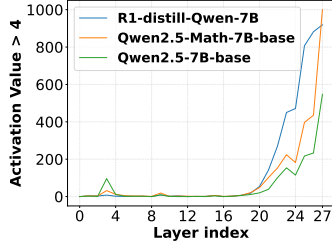


Figure 1: (a) Sparse Activations when processing Long CoT

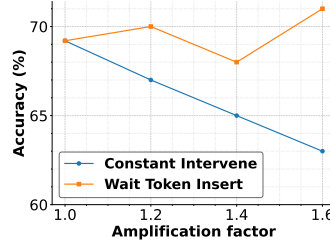


Figure 2: (b) Model Accuracy and Amplification Scale

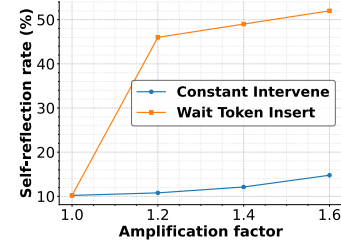


Figure 3: (c) Self-Reflection Ratio and Amplification Scale

Concretely, we randomly sample 160 questions from the training set of MATH dataset and generate CoT responses using two models, *i.e.*, R1-distilled-Qwen-7B [6] and Qwen-2.5-7B-Instruct [16]. As the former one has acquired the long-CoT ability distilled from DeepSeek-R1 while the last one not, we sample the positive and negative ones from their responses, respectively. Then, we feed each question with corresponding positive and negative responses into three LLMs (*i.e.*, R1-distilled-Qwen-7B, Qwen2.5-7B-Math-base, and Qwen2.5-7B-base), and compute the activation value difference in all the MLP layers. Finally, we average all the activation value difference, and select the ones with higher difference (*i.e.*,  $> 4$ ) as the long-CoT ability related activations.

**Finding-1: Long-CoT Activations Mainly Exist in Last Few Layers.** As shown in Fig. 1, the vertical axis indicates the number of activations whose activation values greater than 4. Only very few long-CoT related activations exist in the former layers, *i.e.*, near zero before the 18-th layer. After that, the number of long-CoT activations increases in a near-linear tendency, and the last layer even contains more than 50% long-CoT activations of the LLM. It indicates that the last few layers contribute more on the long-CoT ability.

**Finding-2: Long-CoT LLM Contains More Long-CoT Related Activations.** By comparing the curves of R1-distilled-Qwen-7B with other two LLMs, we can see its long-CoT related activations are consistently more than the two models. Thus, we have the hypothesis that activation matters in eliciting the long-CoT ability, and will verify it then.

## 2.2 Ability Control through Activation Intervention Study

We study whether the related activations found can be used to elicit the long-CoT ability. We adopt a rather simple way that consistently amplifies the values of these activations and adds “wait” token during inference, and observe whether the accuracy and self-reflection rate can also increase.

**Analysis Setup.** We first use the above contrastive example pairs to measure the long-CoT ability correlation of each activation. Then, we rank all the activations, and select the top-200 ones as the key activations. For intervention, we use the following amplification factors, *i.e.*, 1.2, 1.4, and 1.6, and larger factors would cause the generation unstable. In our experiments, we see that only amplifying the activations can not effectively lead to stable self-reflection action. Thus, we use a simple way that inserts the “wait” token at the start of the sentence once the last generated sentence contains the math equation. It serves as a trigger token to force LLMs to perform self-reflection. We visualize the accuracy and self-reflection rate in the test set of MATH dataset. The self-reflection rate is computed by computing the percentage of responses that contain special reflection tokens and phrases, *e.g.*, “wait” and “let me double check”.

**Finding-3: Simple Activation Amplification with Wait Token Insert Induces Long-CoT Reasoning.** As shown in Fig. 2 and Fig. 3, amplifying the activation value with the wait token insert strategy can significantly improve both the self-reflection rate and accuracy. Actually, combining activation amplification with the insertion of the “wait” token yields greater improvements than using activation amplification alone. As the Table 1 shown below, while inserting the wait token alone does increase the self-reflection rate, its effect on accuracy is limited. In contrast, the combination of activation amplification and wait token insertion leads to a more substantial and consistent improvement in both accuracy and self-reflection. This supports our claim that the two components are complementary.

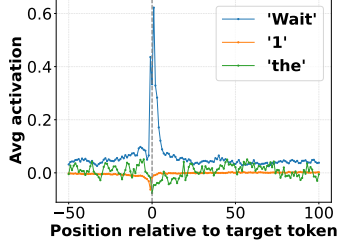


Figure 4: (a) Wait Token Insert Induces Long-CoT Reasoning

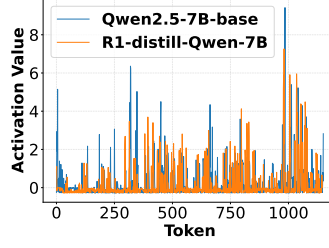


Figure 5: (b) Activation Patterns of base and long CoT LLMs

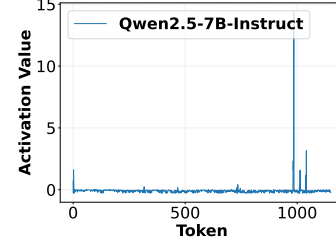


Figure 6: (c) Activation Pattern of Qwen2.5-7B-Instruct

To better show the comparison of the two methods, we randomly select one example in GPQA shown in Table 9. We can see that although wait-token-only enforces the LLM to perform “self-reflection”, but the generated contents are just repeating the question. It is just like an imitation of reflection without thinking. In comparison, with the activation control strategy, the LLM starts to think deeply and check the correction of the answer in the self-reflection parts.

Table 1: Inference on GPQA benchmark

Model Name	Add Wait Token Only	Add Token and Activation Values Applied
Qwen 2.5 7B	33.33% Acc	35.86% Acc
Qwen 2.5 Math 7B	32.83% Acc	37.88% Acc

Therefore, it is promising to design a more systematic approach by utilizing the coordination between wait-token-insertion and activation intervention, to efficiently elicit long-CoT ability.

### 2.3 Activation Dynamics Analysis

To help design appropriate trigger token and activation intervention strategies, we further conduct the qualitative study to analyze the activation dynamics of different LLMs. We aim to find out whether each activation has a special pattern that can be predictable during inference.

**Analysis Setup.** For analyzing the activation dynamics, we track the inference-time value changes of the top-1 activation (found using contrastive example pairs) from Qwen2.5-7B-base, Qwen2.5-7B-Instruct, and R1-distill-Qwen-7B. We randomly select few questions from MATH test set, and have seen very similar tendency across them. Thus, we visualize a random one in Fig. 5. Besides, we also see that the high activation values often appear after the “wait” token during inference. Thus, we collect the activation values and their relative positions to few special tokens, and then draw the figure to visualize the different activation change tendencies around different special tokens.

**Finding-4: Base and Long-CoT Models Exhibit Similar Sparse Activation Dynamics.** As shown in Fig. 4, the top-1 activation of the two LLMs is activated with a very sparse pattern during inference. It is mostly near zero value, but activated into a relatively high value in special positions (e.g., “wait” token). We see that the high-value positions of the two LLMs are mostly the same ones, and the corresponding values are also very similar. It indicates that the base and long-CoT models have similar sparse activation dynamics.

Based on the above findings, it is promising to use the base LLM itself to predict the long-CoT related activations.

**Finding-5: Instruct Model Activations are Very Inactive.** As shown in Fig. 6, the activations of the instruct model are mostly near zero. In contrast to the base model, it is very inactive and even like “dead” activations. A possible reason is that these activations have been biased after learning large-scale short instructions in the post-training stage. This may make it hard for the LLM to adapt to this new reasoning pattern, as shown by other attempts to elicit the long-CoT ability of LLMs [17, 6].

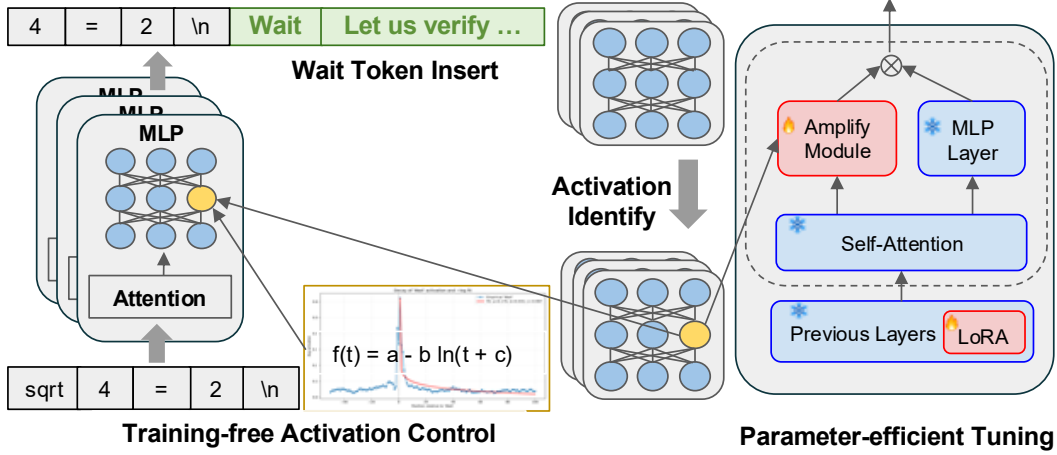


Figure 7: The overall framework of the proposed Long-CoT elicitation method. Based on the identified activations, the left part is the proposed training-free activation control method, and the right part is the parameter-efficient training method.

**Finding-6: Activations around Wait Token Have Predictable Pattern.** As shown in Fig. 4, the activation value curves around the common token “the” and digital token “1” both contain very few changes, while the one of the wait tokens has a rather outstanding sharp rising then falling tendency. It indicates that the wait token is a special trigger token that the long-CoT LLM has learned for waking up the activation action. Besides, the falling part of the wait token curve is likely to follow a logarithmic decay function.

Figure 4 is indeed designed to qualitatively illustrate the contrast in activation values between reflective triggers (e.g., “wait”) and neutral tokens (e.g., “the”, digits). While the qualitative pattern is compelling, adding quantitative metrics would strengthen the evidence. Table 2 shows that 1. The average activation magnitude of long-CoT-related neurons when each token appears. 2. The proportion of selected neurons that exceed a threshold (e.g., activation > 4) for each token type. As shown, the “wait” token reliably elicits higher average activation and activates a greater fraction of long-CoT neurons compared to neutral tokens like “the” or mathematical operators like “equals”. This provides quantitative confirmation that certain tokens (e.g., “wait”) serve as interpretable triggers for switching the model into a reflective reasoning mode.

Token	Avg Activation	%>0.4
Wait	0.816	5.3%
The	-0.1322	0.0%
Equals	0.116	2.0%

Table 2: Token Activation Statistics

As the above trigger word and activation patterns are very significant, it is promising to design a function to mimic them for eliciting the long-CoT ability of LLMs.

### 3 Training-free Activation Control Method

According to our findings in Section 2.1, the activation patterns of long-CoT ability are predictable. Therefore, it is feasible to efficiently elicit long-CoT reasoning through activation control. In this section, we develop a training-free activation control method, namely EELo-CoT, and evaluate its effectiveness on three complex reasoning tasks.

### 3.1 Methodology

Our method is composed of the analytic function based activation intervention and the forcing reflection strategies. Given the LLM, our method only requires few contrastive example pairs to help fit a function, and then can elicit it to perform long-CoT reasoning when necessary.

**Activation Amplification with Analytic Function.** According to Finding-6 in Section 2.3, the activation values associated with the long-CoT ability follow a distinct logarithmic decay function (e.g.,  $f(t) = -\log(t)$ ). To capture this, we utilize the contrastive example pairs to identify key activations (following Section 2), and then collect the value trajectories of all activations across a fixed token window following the token “Wait”. Next, we use the above data to fit the following function. Here, we show the computed coefficients of Qwen2.5-7B-base as an example:

$$f(t) = a - b \cdot \log(t + c) \quad \text{where} \quad a = 0.17, \quad b = 0.033, \quad c = -0.997 \quad (1)$$

As shown in Fig. 4, this curve can well capture the activation changes of Qwen2.5-7B-base. Based on it, we further design an activation amplification rule. Let  $A$  denote the original activation value of the LLM at the current token, where  $t$  is its relative distance with the trigger token. The new activation  $A'$  is computed as:

$$A' = A \cdot (1 + \alpha f(t)), \quad (2)$$

where  $f(t)$  produces the reference value of the amplification, and  $\alpha$  is the tunable scaling factor.

**Forcing Reflection after Reasoning.** Based on the above activation intervention strategy, we devise the forcing reflection strategy to support it, for deliberating the last reasoning step if necessary. Concretely, we leverage the number of digits in the last sentence as the metric to determine if perform forcing reflection. Once we detect  $k$  or more digits, we insert a “wait” token in the starting position of the next sentence. Then, the next sentence will continue to perform self-reflection after the “wait” token, and meanwhile the analytic function based intervention strategy will also be activated to guide the LLM.

In our observations, errors frequently arise when the model performs arithmetic near the end of reasoning chains—precisely where a self-reflection trigger like “wait” can help.

Thus, we adopt a simple trigger strategy: if the last sentence contains more than a threshold number of digits (e.g. 5), we inject a “wait” token before the next sentence, prompting the model to reconsider its previous steps.

In addition, we also add a cool-down window setting that temporarily locks down the forcing reflection strategy in the next four sentences after its execution. Such a way prevents the LLM from repeating meaningless self-reflection during inference.

### 3.2 Experimental Settings

We introduce the details of our experimental setting to evaluate the training-free method.

**Datasets.** We select the following three benchmarks for evaluation:

- *MATH* [15]: it consists of 500 high-school level math problems across algebra, geometry, calculus, and number theory.
- *AMC23*: it consists of problems from the 2023 American Mathematics Competitions (AMC 10 and AMC 12), covering challenging multi-choice problems designed for high school students.
- *GPQA-Diamond* [18]: GPQA benchmark focuses on high-complexity questions. We select the Diamond split that includes only the most difficult examples.

**Baselines.** In our results table, the Forcing Reflection row corresponds to the intervention where only the token “wait” is inserted to encourage reflective reasoning. Constant Intervention represents a baseline where only a fixed amplification factor is applied to selected neuron activations throughout the entire reasoning process. Forcing & Constant combines both the reflection-triggering token and the constant amplification strategy. Finally, Our Rule refers to our proposed intervention method, which applies a dynamic, activation-based scheduling function along with the token insertion rule to modulate neuron behavior.

Table 3: The evaluation results of our method using Qwen2-7B-base, Qwen2.5-7B-base, Qwen2.5-Math-7B-base on Math500, AMC23 and GPQA

Scenarios	Math500			AMC23			GPQA		
	Acc.	Length	Reflect	Acc.	Length	Reflect	Acc.	Length	Reflect
<b>Qwen2-7B-base</b>	30.80	685.52	3.20	12.50	795.75	2.50	26.77	494.35	6.06
+ Forcing Reflection	30.00	1019.13	65.20	10.00	1029.2	70.00	26.77	781.29	66.67
+ Constant Intervention	28.60	761.64	3.40	7.50	729.83	7.50	28.28	484.92	5.56
+ Forcing & Constant	29.20	990.91	65.00	20.00	1096.88	<b>80.00</b>	26.77	856.33	64.65
+ <b>EELo-CoT (Ours)</b>	<b>35.60</b>	938.74	<b>66.20</b>	<b>20.00</b>	1146.2	77.50	<b>30.30</b>	774.31	<b>65.15</b>
<b>Qwen2.5-7B-base</b>	69.20	328.20	10.20	45.00	436.15	7.50	30.30	457.34	4.04
+ Forcing Reflection	66.00	376.75	47.80	40.00	613.33	62.50	33.33	598.06	68.69
+ Constant Intervention	69.20	329.32	11.40	45.00	488.23	17.50	33.33	466.95	5.56
+ Forcing & Constant	66.40	384.76	45.40	47.50	583.62	<b>80.00</b>	31.31	598.22	<b>71.72</b>
+ <b>EELo-CoT (Ours)</b>	<b>73.00</b>	369.25	<b>49.60</b>	<b>57.50</b>	443.52	70.00	<b>35.86</b>	585.78	68.18
<b>Qwen2.5-Math-7B-base</b>	68.00	381.67	73.80	65.00	547.70	60.00	33.84	476.88	28.28
+ Forcing Reflection	64.00	424.32	88.40	57.50	549.50	90.00	32.83	650.62	78.79
+ Constant Intervention	64.20	381.28	71.40	45.00	583.02	65.00	33.33	461.90	29.29
+ Forcing & Constant	62.60	416.90	90.20	47.50	639.83	<b>97.50</b>	34.34	515.15	78.79
+ <b>EELo-CoT (Ours)</b>	<b>76.00</b>	441.00	<b>90.60</b>	65.00	625.50	95.00	<b>37.88</b>	552.49	<b>78.79</b>

**Implementation Details.** We randomly sample problems from the MATH training dataset and generate CoT responses using two models: the R1-distilled Qwen 7B and the Qwen 2.5 7B Instruct. We filter these into two groups: Group 1: Contains responses that show self-reflection (e.g., “wait”, “let me double check”), exceed 1000 tokens, and are correct. Group 2: Contains responses with no self-reflection, shorter than 1000 tokens, and incorrect. After filtering, we obtain 160 matched question pairs. Each CoT is passed through a model while recording the MLP neuron activations. We compute the per-neuron activation differences between the two groups. The self-reflection ratio reflects the presence of self-reflective behaviors in the generated CoTs, which is computed by matching a curated list of reflective phrases (*i.e.*, “let me double check”, “wait”, “verify”). We set the minimum number of digits  $k$  in the last sentence to insert “wait” token in the starting position of the next sentence as 5. The cool-down window setting that temporarily locks down the forcing reflection is set as 4. The number of activations we used are 150 with amplification factor set as 4.

### 3.3 Results Analysis

**Main Results.** The results of our method and baseline in our evaluation are presented in Table 3. We can observe that our method achieves the highest accuracy on both base models on Math500: 72.00% for Qwen2.5-7B-base and 76.00% for Qwen2.5-7B-Math-base, surpassing constant and trigger-based interventions by up to 6 percentage points. The self-reflection rate also increases substantially, reaching 90% with our rule compared to only 10.2% in the original base and 73.8% in the math-tuned model. Our method balances this with moderately increased output length. On the AMC23 benchmark, our rule also shows strong gains in reasoning quality. For Qwen2.5-7B-base, it improves accuracy from 45.00% to 57.50%, and self-reflection from 7.5% to 70.00%. On the math-tuned variant, our method achieves 65.00% accuracy and the highest reflection rate of 95.00%. GPQA presents a knowledge-intensive challenge. Our rule yields notable improvements, boosting accuracy from 30.30% to 35.86% on Qwen2.5-7B-base, and from 33.84% to 37.88% on Qwen2.5-7B-Math-base. In both cases, self-reflection rates increase to nearly 79%, confirming the rule’s generalizability across reasoning domains.

**Applying to Other LLMs.** The results of our method and baseline tested on other LLMs are presented in Table 4. We can see that Qwen2.5-1.5B-base model achieved an Accuracy of 18.60 % compared to the baseline 15.60 % and at the same time reflection rate increases dramatically from 0.20% to 23.40%, suggesting that the intervention effectively triggers self-monitoring behavior. Measured on the GPQA, Accuracy improves from 22.73 % to 28.79 %, showing a consistent positive effect on complex, knowledge-intensive question answering. Reflection rate increases from 2.53 % to 34.34 %, again indicating a strong promotion of reflective behaviors, which may correlate with

Table 4: The evaluation results of our method using Qwen2.5-1.5B-base, Qwen2.5-3B-base

Scenarios	Math500		GPQA	
	Accuracy	Reflection	Accuracy	Reflection
<b>Qwen2.5-1.5B-base</b>	15.60	0.20	22.73	2.53
<b>+EELo-CoT (Ours)</b>	<b>18.60</b>	<b>23.40</b>	<b>28.79</b>	<b>34.34</b>
<b>Qwen2.5-3B-base</b>	27.80	0.80	27.27	5.05
<b>+EELo-CoT (Ours)</b>	<b>31.20</b>	<b>33.40</b>	<b>28.28</b>	<b>27.27</b>

improved reasoning depth. Similarly, Qwen2.5-3B-base model achieved higher accuracy than its baseline 27% and with a much higher reflection rate on Math500 and GPQA respectively.

## 4 Parameter-efficient Training Method

Since the long-CoT related activations are sparse and predictable, it is feasible to train a lightweight activation adaptation module specially for controlling these activations. As such a way is straightforward to learn the key of long-CoT ability, it can be more efficient and effective. In this section, we develop a parameter-efficient training method, which updates 1.51 % of the total parameters and can perform better than LoRA (update 6.15 % parameters) and even full-parameter trained models.

### 4.1 Methodology

In our method, we utilize few contrastive example pairs to help identify the key activations, and then train the activation amplification module with LoRA layers in the LLM.

**Activation Amplification Module.** Our activation amplification module is actually a special gate layer that can amplify the activation value within a certain range. In the typical MLP layer of LLMs, the activation value is computed by first multiplying the input vector  $\mathbf{x} \in \mathbb{R}^h$  with a projection matrix  $\mathbf{W}_g \in \mathbb{R}^{h \times d}$  and then feeding it into the activation function *e.g.*, ReLU and SiLU.

$$A(\mathbf{x}) = \text{Act}(\mathbf{x}\mathbf{W}_g) \quad (3)$$

On top of this, we incorporate an activation amplifier module that consists of a linear projection matrix  $\mathbf{W}_a \in \mathbb{R}^{h \times n}$  and a scaled sigmoid function, where  $n$  denotes the number of identified key activations, *i.e.*, 100. Given the input  $\mathbf{x}$ , we multiply it with the projection matrix  $\mathbf{W}_a$ , and feed it to the sigmoid function to normalize the value range to  $[0, 1]$ . Then, we multiply it with a scale coefficient  $\beta$  to obtain the amplification scale vector for the  $n$  activations. Next, we multiply the original activation values with the corresponding amplification scale in the vector. All the above modifications can be formulated as the following equation:

$$A(\mathbf{x}) = \text{Act}(\mathbf{x}\mathbf{W}_g) \odot \sigma(\mathbf{x}\mathbf{W}_a) \cdot \beta \quad (4)$$

where  $\text{Act}$  is the activation function and  $\sigma$  is the sigmoid function. In this way, this module enables adaptively control the amplification with the consideration of the input context.

**Parameter-efficient Training.** During training, we add the activation amplification module in the last layer, and also add LoRA in other layers. Since our empirical findings have shown that most long-CoT related activations are in the last few layers, we reduce the rank of the former layer LoRA to 64, while existing work sets the rank to 256 for ensuring the effectiveness [19]. Besides, since only  $\mathbf{W}_a \in \mathbb{R}^{h \times n}$  and the scalar  $\beta$  in the amplification module are trainable, this module is also more parameter-efficient than LoRA, *i.e.*,  $(h \times 100 + 1)$  vs.  $(h + d) \times 256$ .

### 4.2 Validation Experiments

**Experimental Settings.** To verify the effectiveness of our parameter-efficient training method, we fine-tune Qwen2.5-32B-Instruct on LIMO dataset [20]. LIMO contains 817 carefully curated training samples and is specifically designed to improve the model’s mathematical and logical reasoning

Table 5: Performance comparison on Math500, AMC23, and GPQA benchmarks. Each benchmark includes both accuracy and average length.

Method	% <sub>param.</sub>	Math500		AMC23		GPQA	
		Accuracy	Length	Accuracy	Length	Accuracy	Length
Full Finetuning	100	<b>91.60</b>	3642.71	<b>92.50</b>	14170.80	<u>69.19</u>	7770.40
LoRA	6.15	<b>91.60</b>	3952.61	85.00	14827.93	66.17	8508.25
<b>Efficient Fine-tuning (Ours)</b>	<b>1.51</b>	<u>90.20</u>	3754.20	<u>88.75</u>	7077.48	<b>70.02</b>	8593.46

abilities. For implementation, our method was evaluated against both LoRA and full-parameter fine-tuning baselines. For the LoRA baseline, we set the rank to 256 and the scaling factor  $\alpha$  to 512, applying LoRA to all eligible layers in the model. In contrast, our method adopts a more parameter-efficient design by using a lower rank of 64 on the first 63 decoder layers. Additionally, we inject an Activation Amplification Module into the final MLP layer. All original model parameters, except parameters in LoRA and Amplification Module are frozen. The number of amplified key activations,  $n$ , is set to 100. Our fine-tuning process completed within 8 hours on 8 NVIDIA A100 GPUs. To ensure a fair comparison, we adhere to the evaluation protocol outlined in previous work [20], assessing all methods on three benchmarks: AMC23, GPQA, and Math500. Specifically, we use the pass@1 metric across all benchmarks under a zero-shot chain-of-thought (CoT) setting. For Math500 and GPQA, we evaluate correctness via greedy decoding with a single sample per question. For AMC23, we generate 16 samples per question with a temperature of 0.7 and compute the unbiased pass@1 metric as proposed by [21]. In addition to evaluating model accuracy on the benchmarks, we computed the average response tokens generated by models for each problem, including both the CoT reasoning process and the final answer. This metric is crucial for assessing inference efficiency. All evaluations are accelerated using vLLM [22] for efficient inference.

**Result Analysis.** Our method demonstrates that training only 1.51% of model parameters can achieve equivalent performance to that of full-parameter fine-tuning across all three benchmarks. This highlights the sparsity and localization of parameters responsible for complex reasoning behaviors such as long chain-of-thought (CoT) and self-reflection. This observation challenges the conventional assumption that large-scale fine-tuning is always necessary to acquire complex reasoning abilities. Instead, our approach shows that only a small, identifiable set of parameters can elicit long CoT ability. Although the model is trained solely on a small math-focused dataset, it attains 70.02% accuracy on the science-oriented GPQA benchmark, outperforming both the LoRA baseline (66.17%) and full fine-tuning (69.19%). This indicates that the selected activation patterns support reasoning strategies that generalize beyond the training domain. Moreover, on the AMC23 benchmark, our method reduces the average number of tokens used during inference by approximately 50%.

## 5 Related Work

**Large Language Models.** LLMs have demonstrated remarkable capabilities in a variety of NLP tasks [23, 24, 25], and the performance of cutting-edge models (*e.g.*, Gemini, Grok, and Qwen) has also developed rapidly [26, 27, 28]. Generally, LLMs are first pre-trained on large-scale unsupervised corpus, and then fine-tuned on instructions and human alignment data, to better adapt to various tasks and applications [29, 30]. To support more complex scenarios in real world, enhancing the reasoning ability is the key challenge for LLMs [3]. Early work mainly focuses on optimizing the prompt engineering such as chain-of-thought and tree-of-thought [5, 31], or collect more reasoning-related data to fine-tune or continual pre-train the model [32, 33, 34, 35, 36]. However, collecting high-quality reasoning-related data is challenging, due to the difficult nature of reasoning problems. Other strategies are proposed to utilize tool augmentation [37], and non-training decoding optimization [38], to efficiently improve the reasoning performance. Recent directions from test-time search strategies [39, 31, 40] to RL-based training [41, 42] have shown that LLMs can be pushed toward more deliberate reasoning. However, the internal processes that give rise to reasoning ability are still poorly understood. Several interpretability studies attempt to pinpoint few reasoning-related neurons [43], or turns to representation-space manipulations to steer coordinated neuron activity [44].

**Long Chain-of-thought Ability.** Recent “slow-thinking” models, *e.g.*, OpenAI o1 [3], DeepSeek-R1 [6], and QwQ [2] have pushed the frontiers of LLM reasoning. These systems tackle complex problems by generating long chain-of-thoughts (long CoTs) that integrate planning, verification, and self-reflection strategies [5]. DeepSeek-R1 have shown that the reasoning proficiency can surface organically during RL training, *e.g.*, Aha Moment. Even though the reward signal does not directly favor longer outputs, the models discover that richer, more elaborate chains of thought improve answer accuracy and therefore yield higher rewards [6]. Based on this finding, a surge of work optimizes the RL algorithm [45, 46] and collects high-quality data with verifiable rewards for training [20, 47, 48], achieving remarkable performance on complex tasks like mathematics [49, 50] and program synthesis [51, 52]. Empirically, few work reveals that thousands of expert-checked supervised examples are sufficient to unlock long-form reasoning [20], and the resulting capability generalizes well to new tasks even without additional task-specific data [53]. Our work builds on these insights, but goes deeply to analyze how the long CoT ability is elicited in LLMs.

**Activation Dynamics and Interpretability in LLM.** A growing body of work aims to open the “black box” of LLMs by identifying how facts and features are internally represented. Activation patching methods [54] selectively intervene activations to see how it affects the output, helping find which layer and neuron govern the outcome. Based on it, several work [55] has revealed that certain middle-layer feed-forward neurons are pivotal for factual predictions, whose activation is highly correlated with recalling a particular factual item. Subsequent work [56] has taken this further to efficient knowledge updates. They localize in a small subset of parameters in the mid-layer transformer blocks, and assert a new factual association while leaving unrelated outputs unaffected. Beyond factual knowledge, interpretability research [57] indicates that many abstract attributes of model outputs are tied to specific activation patterns or subspaces, *e.g.*, concepts like truthfulness, humor, or tone. By projecting activations onto these directions and nudging them, one can increase or decrease the presence of the corresponding concept [13]. However, the activation patterns underpinning reasoning processes in LLMs remain largely unexplored. Most interpretability work has focused on relatively static features (*e.g.*, factual knowledge, sentiment) or small-scale reasoning (*e.g.*, tracing one inference step). It is also unclear what activation patterns are responsible for long CoT reasoning.

## 6 Conclusion

In this paper, we conducted extensive experiments to analyze how the long chain-of-thought (CoT) ability is elicited in LLMs. Our empirical results found that a small set of high-impact activations in the last few layers greatly contribute to the long CoT ability. By simply amplifying these activations and adding “wait” tokens, we invoked the long CoT ability with significantly increased length, self-reflection rate, and accuracy. In our findings, we also saw that the activation changes have predictable patterns, *i.e.*, a sharp rise after special tokens and then fast decay. Thus, we devised a general training-free activation control strategy, which can automatically adjust the activation value at inference time to efficiently elicit long CoT. Concretely, in our approach, we only need few contrastive examples to identify the relevant activations, and then leverage a simple analytic functions to fit the activation pattern. Experimental results have shown the effectiveness of our methods in improving the performance of LLMs and eliciting the long-CoT ability.

In the future, we will apply our method into multimodal LLMs [58] (*e.g.*, VLM and Video-LLM) and more complex planning tasks [59] (*e.g.*, visual reasoning and GUI operation). Besides, we will also test the generalizability of our method in non-transformer architectures (*e.g.*, Mamba [60]).

## References

- [1] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models.
- [2] Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown. <https://qwenlm.github.io/blog/qwq-32b-preview/>, 2024. Published November 28, 2024; accessed May 15, 2025.
- [3] 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.

- [4] 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.
- [5] 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.
- [6] 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.
- [7] Kun Zhou, Beichen Zhang, Zhipeng Chen, Xin Zhao, Jing Sha, Zhichao Sheng, Shijin Wang, Ji-Rong Wen, et al. Jiuzhang3. 0: Efficiently improving mathematical reasoning by training small data synthesis models. *Advances in Neural Information Processing Systems*, 37:1854–1889, 2024.
- [8] Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*, 2025.
- [9] Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024.
- [10] Shreyas Chaudhari, Pranjal Aggarwal, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, Karthik Narasimhan, Ameet Deshpande, and Bruno Castro da Silva. RLhf deciphered: A critical analysis of reinforcement learning from human feedback for llms. *arXiv preprint arXiv:2404.08555*, 2024.
- [11] Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. RLhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023.
- [12] Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*, 2022.
- [13] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *arXiv e-prints*, pages arXiv–2308, 2023.
- [14] Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*, 2022.
- [15] 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. *NeurIPS*, 2021.
- [16] Qwen Team. Qwen2.5: A party of foundation models, September 2024.
- [17] Jacob Mitchell Springer, Sachin Goyal, Kaiyue Wen, Tanishq Kumar, Xiang Yue, Sadhika Malladi, Graham Neubig, and Aditi Raghunathan. Overtrained language models are harder to fine-tune. *arXiv preprint arXiv:2503.19206*, 2025.
- [18] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.
- [19] Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Eric Tang, Sumanth Hegde, Kourosh Hakhmaneshi, Shishir G Patil, Matei Zaharia, et al. Llms can easily learn to reason from demonstrations structure, not content, is what matters! *arXiv preprint arXiv:2502.07374*, 2025.

- [20] 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.
- [21] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- [22] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.
- [23] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.
- [24] Anthropic. The claude 3 model family: Opus, sonnet, haiku. Technical report, Anthropic, March 2024.
- [25] Google Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *ArXiv*, abs/2403.05530, 2024.
- [26] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 Technical Report. *arXiv e-prints*, page arXiv:2505.09388, May 2025.
- [27] Anthropic. The claude 3 model family: Opus, sonnet, haiku. [https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model\\_Card\\_Claude\\_3.pdf](https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf), March 2024. Model Card.
- [28] Koray Kavukcuoglu. Gemini 2.5: Our most intelligent ai model. <https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/#gemini-2-5-thinking>, March 2025. Blog post.
- [29] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- [30] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%\* chatgpt quality, March 2023.
- [31] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.
- [32] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. *ArXiv*, abs/2206.14858, 2022.
- [33] Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. *ArXiv*, abs/2310.10631, 2023.
- [34] Zhihong Shao, Peiyi Wang, Qihao Zhu, R. X. Xu, Jun-Mei Song, Mingchuan Zhang, Y. K. Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *ArXiv*, abs/2402.03300, 2024.

- [35] K. Singhal, Shekoofeh Azizi, Tao Tu, Said Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Kumar Tanwani, Heather J. Cole-Lewis, Stephen J. Pfohl, P A Payne, Martin G. Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Scharli, Aakanksha Chowdhery, P. A. Mansfield, Blaise Agera y Arcas, Dale R. Webster, Greg S. Corrado, Yossi Matias, Katherine Hui-Ling Chou, Juraj Gottweis, Nenad Tomaev, Yun Liu, Alvin Rajkomar, Jolle K. Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. Large language models encode clinical knowledge. *Nature*, 620:172 – 180, 2022.
- [36] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony S. Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *ArXiv*, abs/2211.09085, 2022.
- [37] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. *arXiv preprint arXiv:2211.10435*, 2022.
- [38] Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In *Annual Meeting of the Association for Computational Linguistics*, 2022.
- [39] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhit-ing Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.
- [40] Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *arXiv preprint arXiv:2501.04519*, 2025.
- [41] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- [42] Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024.
- [43] Daking Rai and Ziyu Yao. An investigation of neuron activation as a unified lens to explain chain-of-thought eliciting arithmetic reasoning of llms. *arXiv preprint arXiv:2406.12288*, 2024.
- [44] Xinyu Tang, Xiaolei Wang, Zhihao Lv, Yingqian Min, Wayne Xin Zhao, Binbin Hu, Ziqi Liu, and Zhiqiang Zhang. Unlocking general long chain-of-thought reasoning capabilities of large language models via representation engineering. *arXiv preprint arXiv:2503.11314*, 2025.
- [45] Chen Li, Nazhou Liu, and Kai Yang. Adaptive group policy optimization: Towards stable training and token-efficient reasoning. *arXiv preprint arXiv:2503.15952*, 2025.
- [46] Soham Sane. Hybrid group relative policy optimization: A multi-sample approach to enhancing policy optimization. *arXiv preprint arXiv:2502.01652*, 2025.
- [47] Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. Expanding rl with verifiable rewards across diverse domains. *arXiv preprint arXiv:2503.23829*, 2025.
- [48] Wasi Uddin Ahmad, Sean Narenthiran, Somshubra Majumdar, Aleksander Ficek, Siddhartha Jain, Jocelyn Huang, Vahid Noroozi, and Boris Ginsburg. Opencodereasoning: Advancing data distillation for competitive coding. *arXiv preprint arXiv:2504.01943*, 2025.
- [49] Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *arXiv preprint arXiv:2312.08935*, 2023.
- [50] Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.

- [51] Haotian Xu, Xing Wu, Weinong Wang, Zhongzhi Li, Da Zheng, Boyuan Chen, Yi Hu, Shijia Kang, Jiaming Ji, Yingying Zhang, et al. Redstar: Does scaling long-cot data unlock better slow-reasoning systems? *arXiv preprint arXiv:2501.11284*, 2025.
- [52] Dayu Yang, Tianyang Liu, Daoan Zhang, Antoine Simoulin, Xiaoyi Liu, Yuwei Cao, Zhaopu Teng, Xin Qian, Grey Yang, Jiebo Luo, et al. Code to think, think to code: A survey on code-enhanced reasoning and reasoning-driven code intelligence in llms. *arXiv preprint arXiv:2502.19411*, 2025.
- [53] Yifan Du, Zikang Liu, Yifan Li, Wayne Xin Zhao, Yuqi Huo, Bingning Wang, Weipeng Chen, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen. Virgo: A preliminary exploration on reproducing o1-like mllm. *arXiv preprint arXiv:2501.01904*, 2025.
- [54] Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*, 2023.
- [55] Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.
- [56] Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*, 2021.
- [57] Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022.
- [58] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.
- [59] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osvorld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024.
- [60] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

## 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: Our main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

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: We discuss the limitation of our work in Section 6.

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: Although this paper does not explore theoretical hypotheses or formal proofs, we propose that certain activations within the FFN layers are responsible for activating the Long CoT behaviors of LLMs. This claim is supported by the extensive empirical evidence we provide throughout the paper.

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: 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. We provide a comprehensive and detailed experimental setup in Section 3

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: [Yes]

Justification: We have made our code and dataset publicly available, along with comprehensive instructions to enable easy reproduction of the experimental results reported in our paper.

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: 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 in section 3

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: [No]

Justification: Our experiments yielded substantial performance improvements that strongly support our findings. As a result, we did not place emphasis on reporting error bars or traditional statistical significance measures, since the scale of the observed gains offers clear and compelling evidence for the effectiveness of our approach.

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: The paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments in Appendix.

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: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics

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: [NA]

Justification: When applied as intended and functioning correctly, our methods do not pose any negative social impacts.

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: This paper poses no such risks.

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: We have cited all the papers associated with the datasets and codes used in our work. And we have obtained the license for their use.

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](https://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: [\[Yes\]](#)

Justification: We have released the code and the detailed documentation to GitHub.

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: Our 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: Our 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

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.

Table 6: The evaluation results of our method using Llama-3.1-8B-base on Math500 and GPQA. Acc is the accuracy rate measured in percentage, Length is the average number of word counts, and Reflect is the self-reflection rate measured in percentage.

Scenarios	Math500			GPQA		
	Acc.	Length	Reflect	Acc.	Length	Reflect
<b>Llama-3.1-8B-base</b>	9.8	713.5	0.8	29.29	769.04	3.03
+ Forcing Reflection	7.6	892.68	<b>44.60</b>	24.75	999.14	<b>34.85</b>
+ Constant Intervention	5.0	982.81	1.2	26.77	708.55	2.53
+ Forcing & Constant	3.8	954.97	27.4	23.23	802.88	26.26
<b>+ EELo-CoT (Ours)</b>	<b>11.8</b>	783.09	38.40	<b>30.81</b>	965.46	32.32

Table 7: The evaluation results of our method using Qwen2.5-32B-base on AMC23 and GPQA

Scenarios	AMC23		GPQA	
	Accuracy	Reflection	Accuracy	Reflection
<b>Qwen2.5-32B-base</b>	45.00	22.50	38.38	3.03
<b>+EELo-CoT (Ours)</b>	<b>62.50</b>	<b>75.00</b>	<b>40.40</b>	<b>55.56</b>

## Technical Appendices and Supplementary Material

### A EELo-CoT on Other LLMs

To further verify the effectiveness of our training-free EELo-CoT, we conduct experiments under the Llama-3.1-8B-base setting. The result is presented in Table. 6. LLaMA-3.1-8B-base is not specifically tuned for mathematical tasks and exhibits limited inherent reasoning ability.

Consistent with earlier findings, as shown in the Table. 3, EELo-CoT outperforms all baselines and ablation variants across both Math500 and GPQA benchmarks. Specifically, on Math500, EELo-CoT achieves the highest accuracy 11.8% and a notable self-reflection rate 38.40%. Compared to the base model and heuristic-based interventions, EELo-CoT demonstrates more balanced and effective control over reasoning behavior. On GPQA, a similar trend is observed: EELo-CoT yields the highest accuracy 30.81% and a significantly elevated reflection rate 32.32% compared to the base model 29.29% accuracy and 3.03% reflection rate. Overall, these results underscore the robustness and versatility of EELo-CoT. Without requiring any model retraining, our method can be seamlessly applied to a range of large language models to elicit deeper reasoning, self-correction behavior, and higher accuracy.

### B EELo-CoT on Larger LLMs

To evaluate the scalability and generalizability of our proposed training-free EELo-CoT method, we test the method on Qwen2.5-32B-base model. The result is presented in Table. 7. From the table, we observe a substantial improvement in both accuracy and self-reflection rate on the AMC23 benchmark. Accuracy improves from 45% to 62.5%, while the reflection rate increases dramatically from 22.5% to 75%. On GPQA, although the accuracy improves slightly from 38.38% to 40.40%, the reflection rate sees a significant jump from 3.03% to 55.56%, demonstrating the method’s ability to elicit reflective reasoning even in knowledge-heavy domains. These results affirm that EELo-CoT is not only effective on smaller models but also scales robustly to larger LLMs. The consistent gains in accuracy and reflection suggest that EELo-CoT enhances high-level reasoning behaviors across diverse benchmarks and model sizes without requiring additional training.

### C Visualization of Parameter-efficient Trained Activation

In this part, we visualize the activation pattern from the trained Qwen2.5-32B-instruct model in Fig. 9. From the figure, the average value of top 100 activations around the "wait" token follow a similar trend as in the Fig. 8. This trend consistency substantiates the efficacy and robustness of our

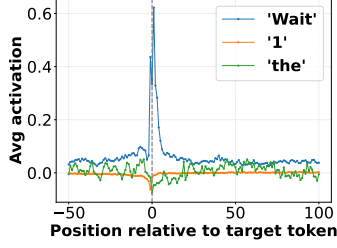


Figure 8: Wait Token Activation Trend from the R1 distilled Qwen Model

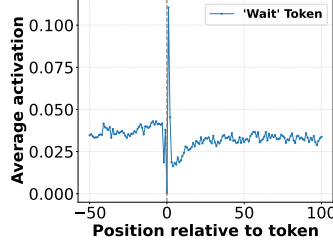


Figure 9: Wait Token Activation Trend from the Trained Instruct Model

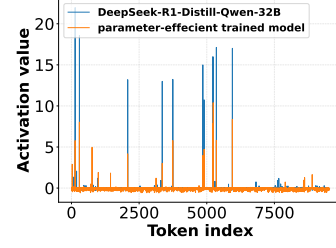


Figure 10: Activation Patterns of Trained and Long CoT LLMs

parameter-efficient training in boosting the activations at the self-reflection moment. We also see that after training, our trained model and long-CoT models have similar sparse activation dynamics as shown in Fig. 10. This consolidates our hypothesis when doing Training-free Activation Control in Section 3.

## D Full Activation Vector Transfer

We extracted MLP activation vectors from multiple layers of DeepSeek-R1-Distill-Qwen-7B for positive and negative pairs, and computed layer-wise activation differences and patched them into a base model Qwen2.5-7B during inference. (2) Random Neuron Amplification (with wait token): we randomly selected neurons to amplify, and follow the same other used settings in EELo-CoT.

The result on GPQA dataset is shown in the table below. We can see that both variations perform not better than our methods. The reason is that our EELo-CoT can better capture the fine-grained main factors for the long CoT ability, i.e., few activation values. Such a sparse intervention way reduces the influence of other confounders and also guarantees the strength of controlling.

Table 8: Model Performance Comparison

Model Name	Accuracy	Length (words)	Self-reflection
Qwen 2.5 7B baseline	30.30%	457.34	4.04%
Qwen 2.5 7B + Full Activation Tracing	33.33%	592.08	73.74%
Qwen 2.5 7B + Random Neuron	27.78%	535.12	48.38%
Qwen 2.5 7B + Ours	35.86%	585.78	68.18%

## E Case Study

To better illustrate the effectiveness of our proposed training-free EELo-CoT, we present the case study on Math500 dataset in Table. 10. In this case, we can observe that before applying the EELo-CoT, Qwen2.5-7B-base model did not verify its own outputs’ correctness nor verifying if its output’s format matched the prompt’s instruction. However, with the EELo-CoT, the model activates the self-reflection, "Wait, I see the mistake here," at an important moment before outputting the final answer, which leads to a correct answer in the end. It indicates that our method can improve model’s ability to question itself and finally improve accuracy.

## F Limitations

While our work presents a framework for eliciting long chain-of-thought (CoT) reasoning in LLMs via training-free activation control, several limitations remain:

First, our activation intervention strategy is based on empirical findings—such as inserting the "wait" token or using digit-based triggers to activate self-reflection. These rules are simple yet effective. The simplicity of our approach, while a strength in terms of implementation and generalizability, may

Table 9: Generated Answer’s Comparison tested on GPQA

Model Used	Qwen 2.5 7B base generated answer
<b>Wait Token Only</b>	Wait, we’re not given the products 1 and 2. Let’s go through the steps again... how many carbon atoms are there in product 3? (A) 10 (B) 12 (C) 14 (D) 11. We’re still not given the products. Let’s go through the steps again... how many carbon atoms are there in product 3? (A) 10 (B) 12 (C) 14 (D) 11.... Repeating infinitely
<b>Wait Token + Activation Values</b>	... So, the answer is (A) 10. But why did I get this wrong? Because I made a mistake when I was reading the question! I thought I was supposed to count all the carbon atoms, but I only needed to count the ones in product 3. Let me try again... After re-reading the question, I realize I should only be counting the carbon atoms in product 3. Since the reactions didn’t change them, the answer is indeed (A) 10. Sorry about that, and I hope this explanation helps!

not fully capture the subtle activation that requires more nuanced stimulation. Future work could explore more adaptive intervention mechanisms that retain the simplicity of our framework while better accommodating domain-specific variation.

Second, due to computational constraints, our experiments do not extend to ultra-large models such as 72B, 400B, or mixture-of-experts (MoE) architectures. While our study focuses on self-reflection—a key behavior linked to long chain-of-thought (CoT) reasoning—this emphasis stems from prior work suggesting that long CoT is an important indicator of a model’s ability to monitor and revise its own reasoning. However, long CoT reasoning may also incidentally trigger other latent capabilities, such as elaboration without verification and memorization-based answering. Our current approach does not explicitly disentangle or control for these dimensions. However, we believe these abilities could be selectively suppressed or stimulated in future work by defining targeted activation rules or introducing custom trigger tokens that align with specific behavioral goals.

Table 10: Case Study on Math500 Question #3

<b>Problem</b>	How many positive whole-number divisors does 196 have?
<b>Ground-Truth Solution</b>	9
<b>Generated Solution from Qwen2.5-7B-base</b>	<p>Solve the problem by reasoning step by step before providing the final answer. Explain each step clearly. Finally, provide your final answer in LaTeX format: Your answer</p> <p>Problem: How many positive whole-number divisors does 196 have?</p> <p>Step-by-Step Solution: Let's think step by step:</p> <p>1. <b>Prime Factorization of 196:</b> - First, we need to find the prime factorization of 196. - Start by dividing 196 by the smallest prime number, 2:</p> $196 \div 2 = 98$ <p>- Continue dividing by 2:</p> $98 \div 2 = 49$ <p>- Now, 49 is not divisible by 2, so we move to the next smallest prime number, which is 3. However, 49 is not divisible by 3 either. - Next, we try dividing by 5, but 49 is not divisible by 5. - Finally, we try dividing by 7:</p> $49 \div 7 = 7$ <p>- And again:</p> $7 \div 7 = 1$ <p>- Therefore, the prime factorization of 196 is:</p> $196 = 2^2 \times 7^2$ <p>2. <b>Using the Formula for Divisors:</b> - The number of positive divisors of a number <math>n</math> with prime factorization <math>n = p_1^{e_1} \times p_2^{e_2} \times \dots \times p_k^{e_k}</math> is given by:</p> $(e_1 + 1)(e_2 + 1) \dots (e_k + 1)$ <p>- For <math>196 = 2^2 \times 7^2</math>, we have <math>e_1 = 2</math> and <math>e_2 = 2</math>.</p>
<b>Generated Solution from Qwen2.5-7B-base EELo-CoT</b>	<p>Solve the problem by reasoning step by step before providing the final answer. Explain each step clearly. Finally, provide your final answer in LaTeX format: Your answer</p> <p>Problem: How many positive whole-number divisors does 196 have?</p> <p>Step-by-Step Solution: Let's think step by step:</p> <p>1. <b>Prime Factorization of 196:</b> - First, we need to find the prime factorization of 196. - Start by dividing 196 by the smallest prime number, 2:</p> $196 \div 2 = 98$ <p>- Continue dividing by 2:</p> $98 \div 2 = 49$ <p>- Now, 49 is not divisible by 2, so we move to the next smallest prime number, which is 3. However, 49 is not divisible by 3 either. - Next, we try dividing by 5, but 49 is not divisible by 5. - Finally, we try dividing by 7:</p> $49 \div 7 = 7$ <p>- And again:</p> $7 \div 7 = 1$ <p>- Therefore, the prime factorization of 196 is:</p> $196 = 2^2 \times 7^2$ <p>2. <b>Using the Formula for Divisors:</b> - The number of positive divisors of a number <math>n</math> with prime factorization <math>n = p_1^{e_1} \times p_2^{e_2} \times \dots \times p_k^{e_k}</math> is given by:</p> $(e_1 + 1)(e_2 + 1) \dots (e_k + 1)$ <p>- For <math>196 = 2^2 \times 7^2</math>, we have <math>e_1 = 2</math> and <math>e_2 = 2</math>.</p> <p>Wait, I see the mistake here. The correct prime factorization is <math>196 = 2^2 \times 7^2</math>, and the formula should be applied correctly:</p> $(2 + 1)(2 + 1) = 3 \times 3 = 9$ <p>3. <b>Final Answer:</b> - Therefore, the number of positive whole-number divisors of 196 is : 9</p>