

---

# Thinkless: LLM Learns When to Think

---

Gongfan Fang Xinyin Ma Xinchao Wang\*

National University of Singapore

gongfan@u.nus.edu, maxinyin@u.nus.edu, xinchao@nus.edu.sg

## Abstract

Reasoning Language Models, capable of extended chain-of-thought reasoning, have demonstrated remarkable performance on tasks requiring complex logical inference. However, applying elaborate reasoning for all queries often results in substantial computational inefficiencies, particularly when many problems admit straightforward solutions. This motivates an open question: Can LLMs learn when to think? To answer this, we propose Thinkless, a learnable framework that empowers an LLM to adaptively select between short-form and long-form reasoning, based on both task complexity and the model’s ability. Thinkless is trained under a reinforcement learning paradigm and employs two control tokens,  $\langle\text{short}\rangle$  for concise responses and  $\langle\text{think}\rangle$  for detailed reasoning. At the core of our method is a Decoupled Group Relative Policy Optimization (DeGRPO) algorithm, which decomposes the learning objective of hybrid reasoning into two components: (1) a control token loss that governs the selection of the reasoning mode, and (2) a response loss that improves the accuracy of the generated answers. This decoupled formulation enables fine-grained control over the contributions of each objective, stabilizing training and effectively preventing the collapse observed in vanilla GRPO. Empirically, on several benchmarks such as Minerva Algebra, MATH-500, and GSM8K, Thinkless is able to reduce the usage of long-chain thinking by 50% - 90%, significantly improving the efficiency of Reasoning Language Models. The code is available at <https://github.com/VainF/Thinkless>

## 1 Introduction

Reasoning Language Models have exhibited notable effectiveness in solving complex tasks that involve multi-hop reasoning and logic-intensive inference. Their capabilities span a range of domains, such as mathematical problem solving [17, 12, 30] and agentic assistants [39, 5]. A primary factor underlying this success is their ability to perform chain-of-thought reasoning [38], wherein intermediate steps are explicitly generated before arriving at a final answer. While this approach is effective for solving challenging problems, applying it uniformly to all queries, regardless of their complexity or the model’s capability can be inefficient. In particular, for questions with straightforward solutions, invoking extended reasoning results in redundant token generation, increased memory footprint, and substantially higher computational cost [11, 7].

In response to these inefficiencies, a growing body of research has sought to enhance the inference efficiency of reasoning models [11, 1, 26, 24, 9, 14, 33]. A prominent direction in this space explores *hybrid reasoning* [4, 36, 2], wherein models dynamically switch between reasoning and non-reasoning modes [41, 2]. Despite promising results, a central challenge persists: determining when a model should engage in elaborate reasoning. Many existing approaches address this by incorporating manually designed heuristics, such as fixed computational budgets [1] or prompt-level control signals like “reasoning on/off” [4, 36]. However, these strategies inherently rely on human prior knowledge and may yield suboptimal or inappropriate control decisions. This underscores a fundamental open

---

\*Corresponding author

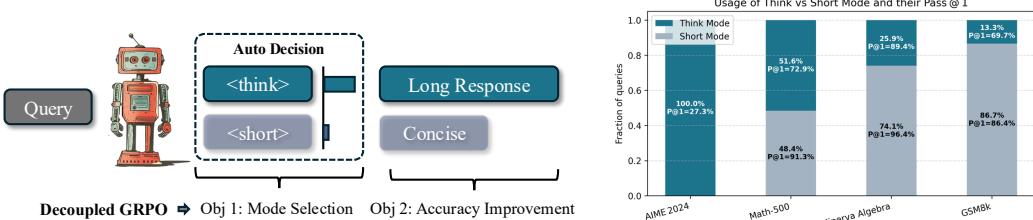


Figure 1: Thinkless learns a hybrid LLM capable of adaptively selecting between thinking and non-thinking inference modes, directed by two special tokens,  $<\text{think}>$  and  $<\text{short}>$ . At the core of our method is a Decoupled Group Relative Policy Optimization, which decomposes and balances the mode selection on the control token and accuracy improvement on the response tokens.

question: Can an LLM *learn* to decide when to think, guided by the complexity of the task and its own capability?

Motivated by this, we explore the fundamental form of hybrid reasoning, where the model is tasked with autonomously deciding whether to generate a short-form or long-form response based on the input query. This decision is guided by three core factors: (1) the complexity of the query, as simpler questions generally merit concise responses, while more intricate ones may necessitate extended reasoning; (2) the capability of the model, since more powerful models are better positioned to employ short reasoning without sacrificing accuracy, whereas less capable models may benefit from longer responses to preserve performance; and (3) the user’s tolerance for the trade-off between efficiency and accuracy, which determines the acceptable level of performance degradation when opting for shorter reasoning. Naturally, reinforcement learning [30, 12, 43] offers a framework to unify these factors, as it allows the model to learn from interactions that reflect both environmental feedback and user-defined preferences. Through iterative exploration and reward-driven updates, the model progressively acquires the ability to make autonomous, context-aware decisions about its reasoning strategy, balancing accuracy and efficiency in a dynamic and data-driven manner.

Building on these insights, we propose Thinkless, a reinforcement learning framework designed to train a hybrid reasoning model capable of selecting between short-form and long-form responses. As illustrated in Figure 3, Thinkless employs two control tokens,  $<\text{think}>$  and  $<\text{short}>$ , which are generated as the first token in the model’s output to signal the intended inference style. The training comprises two stages: a supervised warm-up phase followed by a reinforcement learning phase.

**Distillation for Warm-up.** In the warm-up phase, the model aligns its response style with the designated control tokens via a distillation process. Specifically, it learns to imitate the behavior of two expert models: a reasoning model and a standard instruction-following model, each conditioned on a specific control token ( $<\text{think}>$  or  $<\text{short}>$ ). Additionally, the model is trained on paired long-form and short-form responses for each query, ensuring it can generate both styles with comparable likelihood. This initialization establishes a clear and robust mapping between control tokens and response formats, providing diverse outputs for subsequent reinforcement learning.

**Reinforcement Learning with Decoupled GRPO.** In the reinforcement learning phase, the model is optimized to select the appropriate inference mode based on performance feedback. A natural starting point for this task is the vanilla Group Relative Policy Optimization (GRPO) [12] framework. However, when applied to hybrid reasoning, vanilla GRPO treats all tokens, including the control token and the response tokens uniformly. This introduces a critical imbalance: since the response part often spans hundreds to thousands of tokens and the length of long & short responses varies significantly, the single control token may receive weak and biased gradient signals, ultimately leading to mode collapse at the early stages of training. To this end, we propose a tailored method for hybrid reasoning, termed Decoupled Group Relative Policy Optimization (DeGRPO). As illustrated in Figure 1, DeGRPO explicitly separates the hybrid reasoning objective into two components: (1) Mode Selection, which governs how quickly the policy adapts based on the model’s current accuracy; and (2) Accuracy Improvement, which refines the response content to improve answer correctness under the selected reasoning mode. These two components are inherently interdependent, and effective training requires carefully balancing the learning signals for the control token and the response tokens. For instance, if the mode selection policy adapts too aggressively in favor of long-form reasoning, the

model may ignore short-form responses, resulting in insufficient exploration of the potential of short responses. To this end, DeGRPO assigns distinct weights to the control token and response tokens, promoting a more stable and balanced training dynamic. This design not only mitigates mode collapse but also enables the model to learn both accurate outputs and context-sensitive reasoning strategies. After reinforcement learning, the model learns to accurately identify simple queries and respond using the more efficient non-thinking mode. For instance, on benchmarks such as MATH-500, Minerva Algebra and GSM8K dataset, Thinkless reduces the usage of long-form reasoning by 50% - 90%. And on much more challenging tasks like AIME, the model naturally adopts a higher proportion of long-form reasoning.

In conclusion, this work demonstrates that a Reasoning Language Model can learn when to engage in reasoning before generating a response, guided by the proposed Decoupled GRPO method. This adaptive decision-making substantially reduces inference cost while preserving task performance.

## 2 Related Works

**Efficient Reasoning Models.** Reasoning models generate intermediate steps in a chain-of-thought process before producing a final answer [38, 17]. This paradigm offers significant advantages for tasks involving complex computations, logical deduction, and multi-step reasoning. However, excessively long reasoning chains can lead to substantial computational overhead [7, 9, 11]. To mitigate this, recent research has explored strategies to enhance the efficiency of reasoning models without sacrificing accuracy or generalization [26, 24, 1, 18, 13]. Several techniques, such as reinforcement learning with length penalties [1, 24], supervised fine-tuning using variable-length chain-of-thought data [26], and prompt-based methods [7, 40, 3] have been proposed to encourage concise yet effective reasoning paths. Additionally, latent reasoning techniques aim to encode reasoning steps into compact internal representations, thereby reducing token-level computation while maintaining performance [29]. Parallel efforts in knowledge distillation [16, 42, 27, 20, 6] have facilitated the transfer of reasoning capabilities from large to smaller models, while efficient decoding strategies such as predictive decoding and self-consistency optimization have also yielded notable improvements in inference speed and resource utilization [12, 19, 44].

**Hybrid Reasoning.** While prior work has largely focused on compressing reasoning paths to reduce token generation, an alternative path to efficiency is hybrid reasoning, which dynamically adapts the appropriate inference behaviour based on task complexity [2]. This approach allows models to flexibly alternate between short-form responses and long-chain reasoning as needed. Hybrid reasoning can be realized either through collaborative systems involving multiple models [28, 21] or within a single unified model [2, 36, 4]. In multi-model frameworks, routing mechanisms [28] or speculative decoding techniques [21] are commonly employed. For example, a lightweight model may generate a preliminary answer that a larger model verifies or refines. In contrast, unified models are trained to support both reasoning modes and can switch between them via prompt-based control. Some models adopt fixed prompt formats such as “reasoning on/off” to modulate reasoning depth [4]. However, most existing approaches depend on manually crafted heuristics to balance efficiency and performance. In this work, we explore a learning-based alternative, enabling an LLM to automatically determine its inference behavior based on inputs, without relying on manual control.

## 3 Method

The proposed Thinkless is implemented in two stages: (1) **Distillation for Warm-up**, where we fine-tune a pre-trained reasoning model to unify two reasoning styles, and (2) **Reinforcement Learning with DeGRPO**, where the model is trained under a decomposed objective to select the appropriate reasoning mode while improving the response quality.

### 3.1 Distillation for Warm-up

The first step in our framework is to construct a model  $\pi_\theta$  capable of generating both short- and long-form responses. We leverage two pre-trained experts for distillation: a reasoning model  $\pi_{\text{think}}$ , trained to produce detailed chains of thought via step-by-step reasoning; and an instruction-following model  $\pi_{\text{short}}$ , optimized for generating concise answers aligned with user intent.

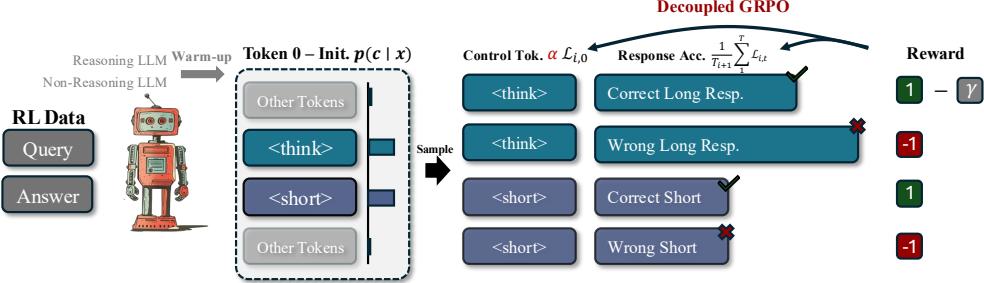


Figure 2: ThinkLess trains a hybrid model that adaptively selects reasoning modes based on task complexity and model capacity. The process begins with distillation, enabling the model to follow control tokens (`<think>` or `<short>`) for guided reasoning. This is followed by reinforcement learning using Decoupled GRPO, which separates training into two objectives: optimizing the control token for effective mode selection and refining the response to improve answer accuracy.

Given a prompt corpus  $\mathcal{X} = \{x_i\}_{i=1}^N$ , we use these models to generate a synthetic paired dataset:

$$\mathcal{D}_{\text{distill}} = \{(x_i, \langle \text{think} \rangle a_i^{\text{think}}, \langle \text{short} \rangle a_i^{\text{short}})\}_{i=1}^N,$$

where  $a_i^{\text{think}} = \pi_{\text{long}}(x_i)$  and  $a_i^{\text{short}} = \pi_{\text{short}}(x_i)$ . Each response is prefixed with a control token  $c \in \mathcal{C} = \{\langle \text{short} \rangle, \langle \text{think} \rangle\}$  that conditions the model on the intended reasoning style. We then fine-tune the target reasoning model  $\pi_\theta$  on this dataset via supervised fine-tuning (SFT). The objective is to learn a multi-style response distribution conditioned on the control token. This distillation phase ensures that the model is capable of generating both types of responses with high fidelity. Moreover, the paired construction of  $\mathcal{D}_{\text{distill}}$  ensures that, the model’s response distribution will be balanced. This helps the follow-up RL process to explore different solutions.

### 3.2 Learning When to Think via Decoupled GRPO

After the distillation phase, the model can produce both long- and short-form answers. what it still lacks is a mechanism for *deciding* which reasoning mode suits a particular input  $x$ . To supply this capability we frame mode selection as a reinforcement-learning problem and optimize a policy  $\pi_\theta(c, a | x) = \pi_\theta(c | x) \pi_\theta(a | x, c)$ , where the first token  $c \in \mathcal{C} = \{\langle \text{short} \rangle, \langle \text{think} \rangle\}$  serves as a *control token* that determines the reasoning mode, and the subsequent tokens  $(a_{i,1}, \dots, a_{i,T_i})$  constitute the generated response. For notational convenience, we denote the entire sequence of the  $i$ -th sample of length  $T_i + 1$  as  $a_i = (a_{i,0}, \dots, a_{i,T_i})$ , where  $a_{i,0} \in \mathcal{C}$  is the control token.

**Reward Design.** Let  $y^*$  denote the ground-truth answer corresponding to the input  $x$ . We consider a minimally designed reward function  $r(a, y^*, c)$ , which assigns different reward values as follows:

$$r(a, y^*, c) = \begin{cases} 1.0, & \text{if } c = \langle \text{short} \rangle \text{ and } \text{Extract-Answer}(a) = y^*, \\ 1.0 - \gamma, & \text{if } c = \langle \text{think} \rangle \text{ and } \text{Extract-Answer}(a) = y^*, \\ -1.0, & \text{if } \text{Extract-Answer}(a) \neq y^*, \end{cases}$$

where the  $1 > \gamma > 0$  introduces preference to the short correct answer over long responses.

**Decoupled Policy Optimization.** Based on the simple reward function, we adopt a GRPO-based framework [30, 23] for training. Let  $\{a_i\}_{i=1}^G$  denote a mini-batch of trajectories sampled from the current policy  $\pi_{\theta_{\text{old}}}$ . The objective is defined as:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x, a_i} \left[ \frac{1}{G} \sum_{i=1}^G \left( \frac{1}{T_i+1} \sum_{t=0}^{T_i} \mathcal{L}_{i,t}(\theta) - \beta \mathbb{D}_{\text{KL}}[\pi_\theta(\cdot | x) \| \pi_{\text{ref}}(\cdot | x)] \right) \right], \quad (1)$$

where  $\mathcal{L}_{i,t}(\theta)$  denotes the token-level surrogate loss formally given by:

$$\mathcal{L}_{i,t}(\theta) = \min \left( \frac{\pi_\theta(a_{i,t} | x, a_{i,<t})}{\pi_{\theta_{\text{old}}}(a_{i,t} | x, a_{i,<t})} \hat{A}_{i,t}, \text{clip} \left( \frac{\pi_\theta(a_{i,t} | x, a_{i,<t})}{\pi_{\theta_{\text{old}}}(a_{i,t} | x, a_{i,<t})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right) \quad (2)$$

In this work, we compute the relative advantage using  $\hat{A}_{i,t} = r - \text{mean}(\mathbf{r})$ , following [23]. This choice is motivated by the observation that our training data contains questions of varying difficulty, which can introduce bias when using standard deviation normalization.

When applied to training, the objective in Equation 1 serves two purposes: (1) to learn an appropriate control token for mode selection, and (2) to improve the accuracy of the response tokens:

$$\frac{1}{T_i + 1} \sum_{t=0}^{T_i} \mathcal{L}_{i,t}(\theta) = \underbrace{\frac{1}{T_i + 1} \mathcal{L}_{i,0}(\theta)}_{\text{Control Token}} + \underbrace{\frac{1}{T_i + 1} \sum_{t=1}^{T_i} \mathcal{L}_{i,t}(\theta)}_{\text{Response Tokens}}. \quad (3)$$

For mode selection, the response style is conditioned on the first token  $a_{i,0}$ , which is trained during the preceding distillation stage. As a result, adjusting the probability of this single control token is sufficient to switch between reasoning modes. Therefore, this token controls the learning of inference mode. For response accuracy, the optimization seeks to improve the generation of the remaining tokens  $a_{i,1:T_i}$ . However, the above Equation 3 introduces two types of imbalance during optimization: (1) *Mode-Accuracy imbalance* - each trajectory contains only one control token but  $T_i$  response tokens, disproportionately reducing the influence of the mode selection compared to the optimization of response accuracy. (2) *Think-Short imbalance* - longer sequences ( $T_i^{\text{think}} \gg T_i^{\text{short}}$ ) further suppress the gradient contribution of the control token due to the normalization factor  $1/(T_i + 1)$ , causing the `<think>` token to be under-optimized compared to the `<short>`. As we will show in the experiment, such imbalance may lead to severe mode-collapse at the beginning of training. To mitigate these imbalances, we propose a decoupled variant of GRPO, denoted as  $\mathcal{J}_{\text{DeGRPO}}$ , which separately normalizes the contributions of the control and response tokens:

$$\mathcal{J}_{\text{DeGRPO}}(\theta) = \mathbb{E}_{x,a_i} \left[ \frac{1}{G} \sum_{i=1}^G \left( \underbrace{\alpha \mathcal{L}_{i,0}(\theta)}_{\text{Control Token}} + \underbrace{\frac{1}{T_i} \sum_{t=1}^{T_i} \mathcal{L}_{i,t}(\theta)}_{\text{Response Tokens}} - \beta \mathbb{D}_{\text{KL}} [\pi_\theta(\cdot | x) \| \pi_{\text{ref}}(\cdot | x)] \right) \right], \quad (4)$$

In DeGRPO, the mode selection  $\mathcal{L}_{i,0}(\theta)$  and response accuracy improvement  $\sum_{t=1}^{T_i} \mathcal{L}_{i,t}(\theta)$  are independently normalized. A length-independent weighting coefficient  $\alpha$  is introduced to balance the optimization between mode selection and response generation. This formulation ensures that the control token receives a consistent gradient scale across both short and long sequences, thereby addressing the mode-mode and think-short imbalance, enabling more stable optimization of reasoning-mode selection. As will be shown in the experiments, an appropriately large  $\alpha$  can make the mode update more efficient. In our experiment, we set  $\alpha = 1/1000$  for stable training.

**Method Summary.** In summary, our method retains the overall structure of the standard GRPO framework. For each query, a mini-batch of samples is drawn from the current policy to estimate the token-level advantages. To address the imbalance between mode selection and response generation, we independently normalize the advantages associated with the control token and response tokens. This separation allows for explicit balancing of their contributions during optimization, leading to more stable and effective training.

## 4 Experiments

### 4.1 Experimental Setups

**LLMs and Datasets.** We employ DeepSeek-R1-Distill-Qwen-1.5B as the base model to train a hybrid reasoning policy. To construct long-short paired responses for the distillation phase, we utilize long-form data from open-source datasets [35, 10] generated by the DeepSeek-R1-671B model [12], which is well-suited for multi-step reasoning. The corresponding short-form answers are derived using Qwen2.5-Math-1.5B-Instruct [41], a compact instruction-tuned model optimized for concise mathematical responses. The hybrid model is directly fine-tuned on this paired dataset via supervised fine-tuning, enabling it to accommodate both long and short reasoning styles. The model is then further optimized using the Decoupled Group Relative Policy Optimization (GRPO) algorithm. For the reinforcement learning stage, we primarily use the DeepScaleR dataset [25], which comprises approximately 40K labeled examples. For evaluation, we mainly focus on math datasets, including AIME [37], Minerva Algebra [15], MATH-500 [22] and GSM-8K [8].

**Training Details.** All experiments were conducted on a single node with 4 H100 GPUs. For the warmup stage, we set the maximum context length to 16K and clip those overlong samples. We

Models	Type	AIME 2024		Minerva Algebra		Math-500		GSM8K	
		Pass@1	#Tokens (Think%)	Pass@1	#Tokens (Think%)	Pass@1	#Tokens (Think%)	Pass@1	#Tokens (Think%)
DeepSeek-R1-1.5B	Base LLM	0.2800	18063	0.9577	3029	0.8608	5675	0.8347	1919
Q-1.5B		0.0200	1300	0.7771	933	0.5168	855	0.7022	466
QMath-1.5B		0.1133	1128	0.9184	586	0.7604	721	0.8572	447
Merging-0.5 [34]	Short CoT	0.1333	8636	0.9292	834	0.7740	1524	0.8332	601
Merging-0.6 [34]		0.1733	10615	0.9321	1091	0.7900	3000	0.8381	747
Merging-0.7 [34]		0.1667	15854	0.9398	1834	0.8108	4347	0.8458	1201
CoT-Valve $\alpha = 8$ [26]		0.2000	10692	0.8079	1903	0.7060	3723	0.7726	773
CoT-Valve $\alpha = 6$ [26]		0.1933	17245	0.9468	2656	0.8024	5167	0.7970	1009
CoT-Valve $\alpha = 4$ [26]		0.2267	17722	0.9439	2965	0.8036	5820	0.8108	1396
L1 <sup>†</sup> [1]		0.2267	3629	0.9452	1566	0.8144	2490	0.8431	602
Router Random	Hybrid	0.1467	8093 (56.00%)	0.9211	1736 (49.28%)	0.7608	3096 (47.92%)	0.8205	1086 (50.99%)
Router Q-7B		0.1667	9296 (46.67%)	0.9250	795 (5.64%)	0.7948	2748 (25.00%)	<b>0.8587</b>	563 (2.35%)
Thinkless		<b>0.2733</b>	7099 (100.00%)	<b>0.9459</b>	1144 (25.88%)	<b>0.8184</b>	2555 (51.56%)	0.8418	624 (13.31%)

Table 1: Empirical results of hybrid reasoning. For hybrid algorithms, we additionally report the proportion of queries executed in the thinking mode during evaluation.  $\dagger$ : For L1, we control the response length by appending “Think for NUM\_TOKENS tokens” to the prompt, where  $NUM\_TOKENS$  was set to match the response length of our model.

train the DeepSeek-R1-Distill-Qwen-1.5B for only 1 epoch. The SFT was conducted on the Megatron framework [32]. For the reinforcement learning stage, we extend the context length to 24K. The model was trained only for 600 steps, using the AdamW optimizer with a learning rate of  $1 \times 10^{-6}$ ,  $\beta = (0.9, 0.999)$ , and a weight decay of 0.01. The batch size is set to 128, with 8 responses sampled for each query, leading to 1024 data points in total. The RL experiments were implemented using the VeRL framework [31]. More details can be found in the Appendix.

## 4.2 Empirical Results on Hybrid Reasoning

**Finding 1.** The learned hybrid reasoning models effectively distinguish complex from simple queries, reducing the use of thinking by 50%–90%.

Table 1 presents a comparison between our method and several existing models or techniques. The first part showcases our baseline model, DeepSeek-R1-Distill-Qwen-1.5B, alongside two instruction-following models designed to generate concise answers. It can be observed that the number of tokens generated by reasoning models is typically 5 to 20 times higher than that of standard models. This highlights the potential for significantly improving efficiency by appropriately selecting the reasoning mode. Notably, on challenging datasets such as AIME and MATH-500, reasoning models tend to outperform others by a large margin. However, on simpler datasets like GSM-8K, extended reasoning offers no clear advantage, and the Qwen2.5-Math-1.5B-Instruct model with only a 4K context length achieves better results.

The second part of the table illustrates techniques for generating shorter chains of thought. One highly effective approach is model merging [34], where DeepSeek-R1-Distill-Qwen-1.5B is interpolated in parameter space with a base model, Qwen2.5-Math-1.5B, to obtain a more efficient model without additional training. Another method is the CoT-Valve [26] technique, which applies supervised fine-tuning (SFT) using LoRA. It allows for controllable reasoning length by adjusting the  $\alpha$  parameter in LoRA to modulate the magnitude of parameter updates. Both methods provide mechanisms for adjusting reasoning length, such as the interpolation ratio and the LoRA  $\alpha$ . To show their performance, we sample outputs across a range of lengths. A notable challenge with these methods is that the optimal reasoning length varies significantly across datasets. Consequently, when a good heuristic, such as merging coefficients of 0.6, is determined on one dataset to reduce token usage, it may result in unexpected performance degradation on other benchmarks, such as AIME.

In the final part of our study, we examine hybrid reasoning strategies, focusing on a comparison with router-based approaches. These methods employ a separate large language model (LLM) to assess query difficulty and dispatch inputs to either a reasoning model or a standard model. While effective to some extent, router models are typically independent and lack comprehensive awareness of the target reasoning models, limiting their ability to make confidence-informed decisions. For instance, on the challenging AIME dataset, where even the reasoning model achieves only 28% accuracy, the router struggles to recognize the its difficulty. In contrast, our method jointly considers both input complexity and model capability, dynamically refining the dispatch strategy through direct interaction with real examples. As a result, it achieves efficient and adaptive reasoning without manual tuning.

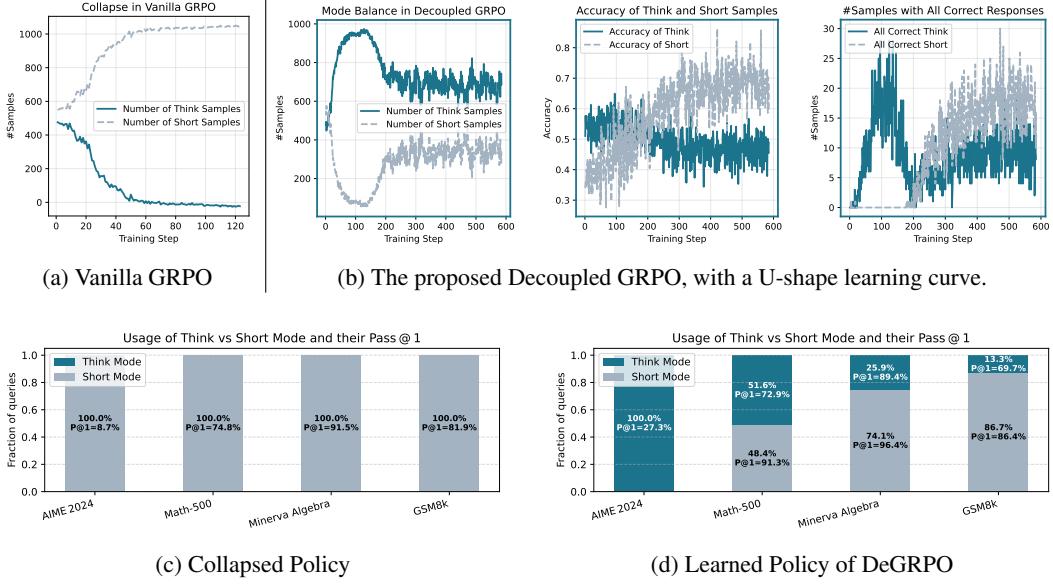


Figure 3: Policy-training comparison between vanilla GRPO and decoupled GRPO.

On the Minerva Algebra dataset, our approach activates the reasoning mode for only 25% of the samples, reducing token usage to one-third of the original, while maintaining performance within a 1% margin. In addition, we found that the RL will also compress the length of long responses, since the algorithm will encourage short and correct answers, producing a gradient towards a more compact response.

### 4.3 Training Dynamics in RL

**Finding 2.** Policy may collapse due to imbalanced update of control tokens in Vanilla GRPO.

**Mode Collapse in RL.** To further analyze how the model learns a reasonable policy, we visualize the training process of RL. Figure 3 (a) illustrates the *Mode Collapse* issue in standard GRPO, where the model develops an excessive preference for either long or short outputs during training. In conventional GRPO, the gradient on the control token is normalized by the total length of the response, which introduces an imbalance between long and short outputs. Specifically, long-chain samples, due to having more tokens, receive slower updates on the `<think>` token, while samples encouraging `<short>` dominate the updates. This imbalance causes the model to collapse rapidly, as shown in Figure 3 (a), the number of generated long-chain responses drops below 10 within just 120 update steps, making it difficult for the model to learn the correct policy. Furthermore, as shown in Figure 3 (c), the model fails to correctly differentiate between samples of varying difficulty, consistently opting for the short-chain reasoning mode.

**Finding 3.** The U-shape learning curve: The proportion of short-chain samples first drops due to low initial accuracy, then rises after accuracy improvement and mode selection take effect.

**The U-Shape Learning Curve.** To mitigate the collapse issue, we propose a Decoupled GRPO algorithm. Figure 3 (b) illustrates the impact of this decoupling on the training process. We observe a characteristic U-shaped curve in the RL process: the proportion of long-chain outputs initially increases and then gradually decreases. After properly balancing the update weights between the control token and response tokens, the model shows a preference for long-chain reasoning in the early stages of training, primarily because long chains tend to yield higher accuracy. As training progresses, we observe an improvement in the accuracy of short-chain responses. This is driven by two factors: (1) reinforcement learning enhances the generation quality, and (2) the model learns to assign simpler queries to the short-chain reasoning mode. As a result, short-chain responses receive increasingly

Model	Mode & Teacher	AIME 2024		Minerva Algebra		Math-500		GSM8K	
		Pass@1	#Tokens	Pass@1	#Tokens	Pass@1	#Tokens	Pass@1	#Tokens
Qwen2.5-1.5B-Instruct	Short (Base)	0.0200	1300	0.7771	933	0.5168	855	0.7022	466
Qwen2.5-Math-1.5B-Instruct	Short (Base)	0.1133	1128	0.9184	586	0.7604	721	0.8572	447
DeepSeek-R1-1.5B	Long (Base)	0.2800	18063	0.9577	3029	0.8608	5675	0.8347	1919
<b>R1-1.5B + OpenR1-97K</b>	◆ Long (R1-671B) ◇ Short (QMath-1.5B)	0.2933 0.0733	18880 3023	0.9456 0.9001	3239 642	0.8336 0.7212	6780 1004	<b>0.8382</b> 0.7985	3423 433
<b>R1-1.5B + OpenThoughts-114K</b>	◆ Long (R1-671B) ◇ Short (QMath-1.5B)	0.2600 0.0800	17964 3807	0.9506 0.8947	3122 702	0.8280 0.7136	6430 1004	0.8039 0.7886	2427 449
<b>R1-1.5B + OpenThoughts-1M</b>	◆ Long (R1-671B) ◇ Short (QMath-1.5B)	<b>0.3067</b> <b>0.0800</b>	18685 3977	<b>0.9530</b> <b>0.9051</b>	3064 690	<b>0.8360</b> <b>0.7276</b>	6209 989	0.8253 <b>0.8202</b>	2363 441

Table 2: The effectiveness of different SFT datasets during the warm-up stage. Since these models have not yet been optimized via reinforcement learning, the control tokens `<think>` and `<short>` are manually inserted to elicit the desired response patterns.

higher rewards, encouraging the model to further explore the feasibility of short reasoning. This behavior is manifested in the rising proportion of short-chain outputs over time, with many short responses achieving perfect accuracy in the later stages of training. Additionally, we observe a decline in the accuracy of long-chain responses during the latter half of training. This phenomenon is not due to a degradation of the model’s reasoning ability but rather a shift in task allocation: more difficult queries, which cannot be solved via short reasoning, are assigned to the long-chain mode, thereby lowering its average accuracy.

**Finding 4.** The weight  $\alpha$  of the control token governs the learning speed of mode selection.

**The Influence of Decoupling.** Building upon the above analysis, we further visualize the effect of decoupling on model behavior. In Figure 4, we show the number of samples that are correctly answered with short responses across all sampled trajectories. We compare the results under two settings: a high control token update weight (0.5) and the weight used in our method (0.001). We observe that with a higher weight on the control token, the all-correct short samples emerge earlier in training. This is because the control token is updated more aggressively, allowing the model to focus more on learning the mode selection. However, excessively fast policy updates can be problematic. For example, some samples may initially yield low accuracy under short responses, but reinforcement learning could eventually improve their short-mode performance. If the model’s strategy updates too quickly, it tends to assign such samples prematurely to the long-chain mode, degenerating the algorithm to a simple binary classifier based on initial accuracy rather than a collaborative learning of mode selection and accuracy improvement.

#### 4.4 Details of Warm-up Distillation

**Finding 5.** Reasoning LLMs are effective short-response learners.

In this work, knowledge distillation is deployed for warm-up, serving as a critical step to equip the model with the basic ability to generate both long chains and short responses. In this section, we provide additional implementation details. Specifically, we consider multiple datasets for distillation. We compare three datasets of increasing scale and domain coverage: (1) OpenR1, a mathematics-only dataset with rigorously verified solutions; (2) OpenThoughts-114K, a compact yet multi-domain dataset labeled by DeepSeek-R1-67B, covering mathematics, science, and programming; and (3) OpenThoughts-1M, a large-scale and diverse collection that subsumes the former two. Table 2 presents the training results on these datasets. Notably, we find that generating short responses using a long-chain reasoning model is relatively straightforward; even with the smallest dataset, OpenR1-97K, the target model successfully learns to produce short outputs. However, this distillation

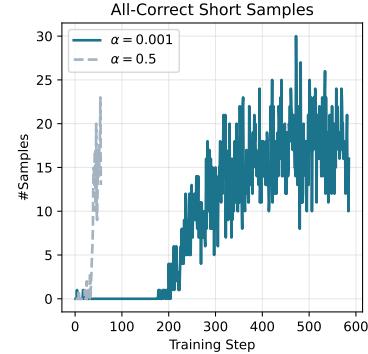


Figure 4: A large token loss coefficient  $\alpha$  accelerates the shift in reasoning behavior, leading to the rapid emergence of all-correct short-mode samples.

$P(<\text{think}>|x) = 1.000000$

Let  $\mathcal{S}$  be the set of points  $(a,b)$  with  $0 \leq a, b \leq 1$  such that the equation  $[x^4 + ax^3 - bx^2 + ax + 1 = 0]$  has at least one real root. Determine the area of the graph of  $\mathcal{S}$ .

$P(<\text{think}>|x) = 0.504883$

Find the projection of  $\mathbf{a}$  onto  $\mathbf{b} = \begin{pmatrix} 2 & 6 & 3 \end{pmatrix}$  if  $\mathbf{a} \cdot \mathbf{b} = 8$ .

$P(<\text{think}>|x) = 0.003534$

The arithmetic mean of 7, 2,  $x$  and 10 is 9. What is the value of  $x$ ?

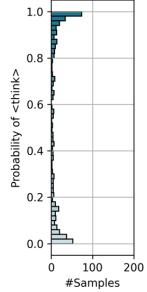


Figure 5: Distribution of the model’s probability of emitting  $<\text{think}>$  on MATH-500. The samples with the highest, medium, and lowest probabilities are highlighted. The example with almost 0 thinking score mainly involves straightforward computation, and the query with 1.0 probability relies more on understanding and logical reasoning. More examples and LLM responses can be found in the appendix.

Model	Easy Problem	Hard Problem
<b>R1-1.5B</b>	0.9163 (Think)	0.5977 (Think)
<b>Qwen-1.5B</b>	0.7531 (Short)	0.2835 (Short)
<b>Thinkless-1.5B</b>	0.9079 (Short)	0.6935 (Think)

(a) Performance on easy and hard problems

Mode	Length Mean	Length Std	Acc
<b>Think Mode</b>	3185	2446	0.8436
<b>Short Mode</b>	672	465	0.7576

(b) The output length and accuracy

Table 3: The statistics of think and short modes on the MATH-500 dataset.

process may incur some performance degradation. For instance, on the Math-500 benchmark, the accuracy of the distilled hybrid model is slightly lower than that of the original model, which may affect the downstream performance during the RL phase. For the distillation stage, larger and more comprehensive datasets can lead to improved performance. However, the marginal gains diminish as the dataset size increases. For example, expanding the dataset from 14K to 1M results in only a 1% improvement in long-chain accuracy on the Math-500 benchmark. This work provides a preliminary validation of the effectiveness of simple distillation, and we leave the construction of stronger initial hybrid models as an important direction for future research.

## 4.5 Case Study

**Finding 6.** The model adaptively adjusts reasoning depth to task difficulty.

**Visualization.** Figure 5 presents a case study on the model’s predicted probability of selecting the  $<\text{think}>$  token across the MATH-500 dataset. The distribution reveals that the model makes smooth and hierarchical predictions for queries of different difficulty. In addition, we highlight representative examples corresponding to high, medium, and low confidence levels to illustrate the model’s decision behavior. It can be observed that samples assigned to the short reasoning mode are typically simple arithmetic problems that do not require deep or complex reasoning. In contrast, questions routed to the thinking mode tend to be more complex, involving multiple conditions and concepts. Overall, the results reflect a well-calibrated policy that adapts reasoning depth based on task complexity.

**Mode Statistics.** To analyze the learned policy, we categorize the samples in the Math-500 dataset into easy and hard problems based on the decisions made by our model. As shown in Table 3a, we evaluate the performance of the base models (DeepSeek-R1-1.5B [12] and Qwen2.5-1.5B [41]) on these subsets. The model indeed learns to distinguish problem difficulty. At the same time, the RL process explores the potential of short responses across various problems. This results in a significant performance improvement on easy problems, increasing accuracy from 0.7531 to 0.9079. Further, we compared the relative output lengths of the model given the same prompt in Table 3. Across 2,500 trials on Math-500, we found that in 99.8% of cases, the output generated in Think Mode was longer than that in Short Mode.

## 5 Limitations and Future Works

This work presents an effective reinforcement learning framework that enables a hybrid model to adapt its inference mode based on both problem complexity and its own capabilities. However, several limitations remain. For instance, during the warm-up phase, we only validate a simple supervised fine-tuning (SFT) approach without extensive parameter tuning to achieve optimal performance, which results in a slight performance drop in the initial model for reinforcement learning. Exploring better strategies for constructing the hybrid model, such as merging techniques or lightweight fine-tuning methods like LoRA to mitigate catastrophic forgetting, could further enhance the overall performance. In addition, while our algorithm has been validated on DeepScaleR, a dataset containing 40K mathematical problems, future work could expand to a broader range of datasets, incorporating more diverse domains to enable more general and practical hybrid reasoning capabilities.

## 6 Conclusion

This paper proposes a reinforcement learning framework for building a hybrid reasoning model. It autonomously decides whether to generate a short response or engage in long-form reasoning based on the complexity of the input. The core of our approach is a Decoupled GRPO algorithm, which separates the reinforcement learning objective into two components: mode selection on the control token and accuracy improvement on the response tokens. This decoupling enables a more balanced contribution between the two learning objectives. Our method effectively reduces unnecessary long-form reasoning, thereby lowering overall system cost and improving user latency.

## Acknowledgements

This project is supported by the National Research Foundation, Singapore, and Cyber Security Agency of Singapore under its National Cybersecurity R&D Programme and CyberSG R&D Cyber Research Programme Office (Award: CRPO-GC1-NTU-002).

## References

- [1] Pranjal Aggarwal and Sean Welleck. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*, 2025.
- [2] Anthropic. Claude 3.7 Sonnet. <https://www.anthropic.com/clause/sonnet>, 2025. Accessed: 2025-05-10.
- [3] 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.
- [4] Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, Gerald Shen, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Yoshi Suhara, Olivier Delalleau, Zijia Chen, Zhilin Wang, David Mosalanezhad, Adi Renduchintala, Haifeng Qian, Dima Rekesh, Fei Jia, Somshubra Majumdar, Vahid Noroozi, Wasi Uddin Ahmad, Sean Narenthiran, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Igor Gitman, Ivan Moshkov, Wei Du, Shubham Toshniwal, George Armstrong, Branislav Kisacanin, Matvei Novikov, Daria Gitman, Evelina Bakhturina, Jane Polak Scowcroft, John Kamalu, Dan Su, Kezhi Kong, Markus Kliegl, Rabeeh Karimi, Ying Lin, Sanjeev Satheesh, Jupinder Parmar, Pritam Gundecha, Brandon Norick, Joseph Jennings, Shrimai Prabhuloye, Syeda Nahida Akter, Mostofa Patwary, Abhinav Khattar, Deepak Narayanan, Roger Waleffe, Jimmy Zhang, Bor-Yiing Su, Guyue Huang, Terry Kong, Parth Chadha, Sahil Jain, Christine Harvey, Elad Segal, Jining Huang, Sergey Kashirsky, Robert McQueen, Izzy Puttermann, George Lam, Arun Venkatesan, Sherry Wu, Vinh Nguyen, Manoj Kilaru, Andrew Wang, Anna Warno, Abhilash Somasamudramath, Sandip Bhaskar, Maka Dong, Nave Assaf, Shahar Mor, Omer Ullman Argov, Scot Junkin, Oleksandr Romanenko, Pedro Larroy, Monika Katariya, Marco Rovinelli, Viji Balas, Nicholas Edelman, Anahita Bhiwandiwalla, Muthu Subramaniam, Smita Ithape, Karthik Ramamoorthy, Yuting Wu, Suguna Varshini Velury, Omri Almog, Joyjit Daw, Denys Fridman, Erick Galinkin, Michael Evans, Katherine Luna, Leon Derczynski, Nikki Pope, Eileen Long, Seth Schneider, Guillermo Siman, Tomasz Grzegorzek, Pablo Ribalta, Monika Katariya, Joey Conway, Trisha Saar, Ann Guan, Krzysztof Pawelec, Shyamala Prayaga, Oleksii Kuchaiev, Boris Ginsburg, Oluwatobi Olabiyi, Kari Briski, Jonathan Cohen, Bryan Catanzaro, Jonah Alben, Yonatan Geifman, Eric Chung, and Chris Alexiuk. Llama-nemotron: Efficient reasoning models, 2025.
- [5] Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, Fan Yang, et al. Research: Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025.
- [6] Xiaoshu Chen, Sihang Zhou, Ke Liang, and Xinwang Liu. Distilling reasoning ability from large language models with adaptive thinking. *arXiv preprint arXiv:2404.09170*, 2024.
- [7] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuseng Zhang, et al. Do not think that much for  $2+3=?$  on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.
- [8] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [9] Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, et al. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*, 2025.
- [10] Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025.
- [11] Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Efficient reasoning models: A survey. *arXiv preprint arXiv:2504.10903*, 2025.
- [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] Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware llm reasoning. *arXiv preprint arXiv:2412.18547*, 2024.
- [14] Masoud Hashemi, Oluwanifemi Bamgbose, Sathwik Tejaswi Madhusudhan, Jishnu Sethumadhavan Nair, Aman Tiwari, and Vikas Yadav. Dna bench: When silence is smarter—benchmarking over-reasoning in reasoning llms. *arXiv preprint arXiv:2503.15793*, 2025.

[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] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[17] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Hel-  
yar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[18] Yu Kang, Xiangui Sun, Liangyu Chen, and Wei Zou. C3ot: Generating shorter chain-of-thought without compromising effectiveness. *arXiv preprint arXiv:2412.11664*, 2024.

[19] Chenglin Li, Qianglong Chen, Liangyue Li, Caiyu Wang, Yicheng Li, Zulong Chen, and Yin Zhang. Mixed distillation helps smaller language model better reasoning. *arXiv preprint arXiv:2312.10730*, 2023.

[20] Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. Small models struggle to learn from strong reasoners. *arXiv preprint arXiv:2502.12143*, 2025.

[21] Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient llm reasoning. *arXiv preprint arXiv:2501.19324*, 2025.

[22] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

[23] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

[24] Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*, 2025.

[25] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. *Notion Blog*, 2025.

[26] Xinyin Ma, Guangnian Wan, Rumpeng Yu, Gongfan Fang, and Xinchao Wang. Cot-valve: Length-compressible chain-of-thought tuning. *arXiv preprint arXiv:2502.09601*, 2025.

[27] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*, 2022.

[28] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

[29] Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. 2025.

[30] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Huawei 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.

[31] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

[32] Mohammad Shoeybi, Mostafa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[33] Gaurav Srivastava, Shuxiang Cao, and Xuan Wang. Towards reasoning ability of small language models. *arXiv preprint arXiv:2502.11569*, 2025.

[34] 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.

- [35] Open Thoughts Team. Open thoughts, January 2025.
- [36] Qwen Team. Qwen3, April 2025.
- [37] Hemish Veeraboina. Aime problem set 1983-2024, 2023.
- [38] 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.
- [39] Junde Wu, Jiayuan Zhu, and Yuyuan Liu. Agentic reasoning: Reasoning llms with tools for the deep research. *arXiv preprint arXiv:2502.04644*, 2025.
- [40] Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025.
- [41] 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.
- [42] Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. Distilling system 2 into system 1. *arXiv preprint arXiv:2407.06023*, 2024.
- [43] Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *arXiv preprint arXiv:2503.18892*, 2025.
- [44] Xunyu Zhu, Jian Li, Can Ma, and Weiping Wang. Improving mathematical reasoning capabilities of small language models via feedback-driven distillation. *arXiv preprint arXiv:2411.14698*, 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: This work proposes a RL method to train hybrid reasoning LLMs, which can reduce the usage of CoT for improved efficiency.

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: The limitations are discussed in Section 5.

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: No theoretical result in this 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: We provided details and hyperparameters in the experiment part.

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 will release all necessary data and code for public access.

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, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

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

Justification: We discussed the settings in each subsection of the experiments.

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: error bars were not reported.

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: 4 H100 GPUs

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

Justification: This work can reduce the inference cost of reasoning LLMs.

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: No such risk.

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: All works were properly cited.

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 discussed the details of generated data in the experiments.

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: 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.
- 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 rigorosity, or originality of the research, declaration is not required.

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

Justification: the core method uses LLM for distillation and RL.

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 Implementation Details

In this work, we adopt a two-stage training framework: we first fine-tune a reasoning model to generate both long and short responses, and then train it to learn a policy for selecting between the two reasoning modes. Figure 6 shows an example of training a 1.5B LLM. In this section, we provide more details and evaluation.

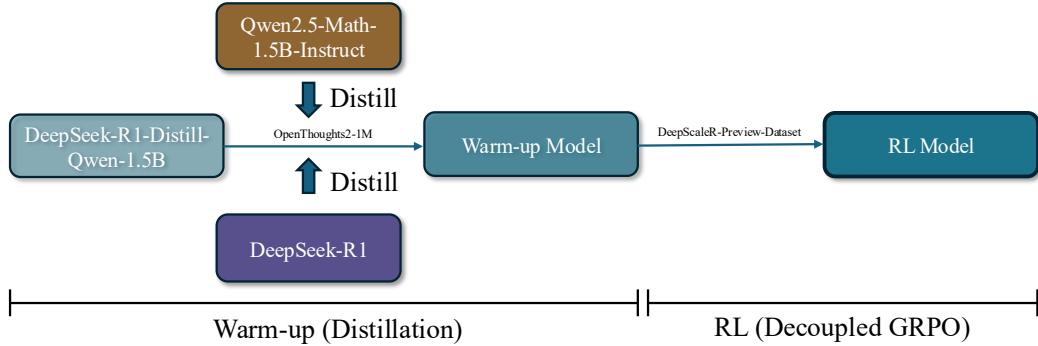


Figure 6: An overview of the SFT and RL training

### A.1 Warmup

We construct an initial Hybrid Reasoning Model through a simple distillation process. The main goal of this process is to establish a connection between control tokens and the model’s response style. To achieve this, we train on samples generated by `Qwen2.5-Math-1.5B-Instruct` and `DeepSeek-R1` 671B. Specifically, we use the `OpenThoughts2-1M` dataset, and for each query, we create paired long and short responses from the two teacher models. For long-chain reasoning responses, the DeepSeek model naturally begins with the special token `<think>`, so its outputs can be used directly. For the responses generated by the `Qwen2.5-Math-1.5B-Instruct` model, we prepend an additional `<short>` as the control token.

During training, we include both the long and short training samples of a given query within each batch. We fine-tune the pre-trained model `DeepSeek-R1-Distill-Qwen-1.5B` using a simple supervised fine-tuning (SFT) approach. The training is conducted with the AdamW( $lr = 1e-5$ , weight decay=0.1, betas = (0.9, 0.95)), and we apply a Cosine Annealing Scheduler for learning rate decay, with a minimum learning rate of  $1e-6$ . The model is trained for a single full epoch on a dataset containing 2 million samples, with a batch size of 128, resulting in approximately 17,000 optimization steps. Gradient clipping is applied with a norm threshold of 1.0. The training is conducted on 4 H100 GPUs, and a Distributed Optimizer is used to reduce memory consumption.

### A.2 Reinforcement Learning

In the reinforcement learning stage, we use the hybrid model obtained from the previous step as the foundation. This model is capable of generating both long and short responses with similar probabilities, which facilitates the exploration of various solution strategies across different types of problems. We adopt a simple reward design 1.0, 0.5, -1.0, -1.0, where a reward of 1.0 is assigned to correct short responses, and a relatively lower reward of 0.5 is given to correct long responses. Incorrect responses, regardless of length, receive a reward of -1.0.

We employ the Decoupled GRPO method proposed in this work to learn the policy, which assigns independent, length-agnostic weights to control tokens. We use a simple weight of 0.001 to balance the contributions from control tokens and response tokens. Notably, this setting is equivalent to standard GRPO training with a fixed output length of exactly 1000 tokens. The algorithm is implemented using the VeRL framework. During training, we use a batch size of 128, sampling 8 responses per query, resulting in a total of 1024 training samples per step. Sampling is performed at a temperature of 0.6. The model is trained with a learning rate of  $1e-6$  for 600 optimization steps.

### A.3 Evaluation

We evaluate our model on four mathematical reasoning benchmarks: AIME [37], Minerva Algebra [15], MATH-500 [22], and GSM8K [8]. The evaluation is conducted using the `lm_eval` framework, with a simple prompting template:

```
Please reason step by step, and put your final answer within
\boxed{.}\n{{problem}}
```

During inference, we adopt a sampling strategy with  $\text{temperature} = 0.6$  and  $\text{top\_p} = 0.95$ . For each problem, we generate 5 completions and report the average accuracy across them. To extract the final answers from LLM responses, we apply a rule-based parsing strategy. We first attempt to extract the value enclosed within `\boxed{}`. If this fails, we fall back to several regular expression patterns to match answers in alternative formats, such as:

```
\*?Final\s+Answer\*?\s*[:=]\s*([0-9.-]+)
```

## B Social Impact

This work does not pose any significant negative societal impact. We propose an adaptive hybrid model that can reduce long-chain reasoning by 50%–90%, significantly lowering inference costs and power consumption, thereby reducing carbon emissions. Moreover, the adaptive reasoning framework improves the responsiveness of AI systems, particularly in real-time applications. This will help improve the user experience of AI products.

## C Case Study

We further present additional case studies demonstrating how the learned LLM adopts different inference modes. In Case 1, the input query is a simple arithmetic problem: “ $1 + 1 = ?$ ”. The Thinkless LLM correctly selects the short reasoning mode and responds succinctly. In contrast, the DeepSeek-R1 model, despite its 671B parameters, engages in an unnecessarily lengthy reasoning process involving multiple revisions, resulting in increased computational cost and user latency.

In Case 2, we consider a more complex query that requires step-by-step logical reasoning. In this scenario, both the Thinkless model and DeepSeek-R1 appropriately generate long-chain reasoning to validate various solution paths, producing 1,494 and 1,868 tokens, respectively.

## Case 1: A Simple Question

Query:  $1+1=?$

Thinkless Response (Short Mode - 99 Tokens)

<short>

To solve the problem  $1 + 1$ , we will follow a step-by-step approach:

1. \*\*Identify the numbers involved\*\*: We have the numbers 1 and 1.
2. \*\*Understand the operation\*\*: The operation we need to perform is addition.
3. \*\*Perform the addition\*\*: When we add 1 and 1 together, we get 2.

So, the final answer is 2.

DeepSeek R1 671B (Think Mode - 376 Tokens)

Okay, so I need to figure out what 1 plus 1 equals. Let me start by thinking about basic addition. When you add two numbers together, you're combining their values. So if I have one apple and someone gives me another apple, how many apples do I have in total? That should be two apples.

Wait, let me make sure I'm not missing something here. In the simplest terms, addition is taking two quantities and putting them together. So 1 plus 1 means combining a single unit with another single unit. If I use my fingers, holding up one finger on my left hand and one on my right, then when I put them together, I have two fingers up. That seems right.

But maybe there's a trick to this question. Sometimes problems that look easy might have a different answer depending on the context. For example, in binary,  $1 + 1$  equals 10. But the question doesn't specify a number system, so I should assume it's in the decimal system, which is the default. So in decimal,  $1 + 1$  is definitely 2.

Another way to check is by using the number line. Starting at 1, if I move one unit to the right, I land on 2. That confirms it. Or using basic arithmetic rules: the sum of 1 and 1 is the next number in the sequence after 1, which is 2.

I guess there's no ambiguity here. Unless the question is trying to be philosophical or something, but mathematically, it's straightforward. So the answer should be 2.

The sum of 1 and 1 is calculated as follows:

$$1 + 1 = 2$$

In decimal arithmetic, adding the numbers 1 and 1 results in the number 2.

<think> Okay, so I have this simple math problem here:  $1 + 1 = ?$  Hmm, that seems straightforward, but I guess I should think about it carefully to make sure I'm not missing anything. Let me try to visualize it.

First, I know that 1 is just one unit or object. So if I have one apple and someone gives me another apple, how many apples do I have now? Well, I should count them: one, two. So that would make it 2 apples. So,  $1 + 1$  equals 2. That seems right. But wait, is there another way to think about this? Maybe using objects or fingers? Let me try using my fingers. If I hold up one finger and then another finger, how many fingers am I up on? One, two. Yep, that's 2 again. So that's consistent with my earlier thought. Alternatively, maybe I can use my knowledge of numerals. I know that in the decimal system, each numeral represents a certain value. The numeral 1 represents one, and when you add another 1, you're just moving one more step in the counting sequence. So 1 comes after 0, and then 1 again, and so on. So,  $1 + 1$  lands me at 2. Wait, let me think about this in terms of quantities. If I have a set of objects, say two apples, and someone removes one apple, how many are left? That would be 1. So, if I had 1 apple and someone gave me another, I'd have 2. That seems to confirm it. Is there a different perspective? Maybe using number lines. If I start at 1 and move one unit to the right on the number line, I end up at 2. So, that's another way to visualize addition.  $1 + 1 = 2$  because moving from 1 to 2 is adding one more. I also remember in basic arithmetic, addition is commutative, meaning the order doesn't matter. So  $1 + 1$  is the same as  $1 + 1$ , which is 2. If I switch them around,  $1 + 1$  is still 2, so that doesn't change anything. What about if I use different representations? Like, in Roman numerals, 1 is I, and 2 is II. So,  $I + I$  would be II, which is 2. That's another way to confirm it. Wait, maybe I can think about it in terms of equations. If I have an equation:  $1 + 1 = x$ . To solve for  $x$ , I can subtract 1 from both sides, so  $x = 1 + 1 - 1$ , which simplifies to  $x = 1$ . But that doesn't make sense because  $1 + 1$  is 2, not 1. Maybe I made a mistake in the algebra. Let me try again. If I have  $1 + 1 = x$ , then subtracting 1 from both sides would give me  $x = 1$ . That can't be right because  $1 + 1$  is definitely 2. So maybe I can't solve it that way. Maybe I need to use a different approach. Alternatively, maybe I can use the concept of sets. If I have one set with one object and another set with one object, combining them gives me two objects. So, the union of two single-element sets has two elements. Therefore,  $1 + 1$  equals 2. I guess another way to think about it is through counting. If I count one thing and then another, I have two in total. So, 1 (first) + 1 (second) = 2 (total). That makes sense. Is there a real-world example that could help solidify this? Sure. If I have a dollar and I add another dollar, I have 2. So, 1 dollar plus 1 dollar equals 2 dollars. That's a practical application of addition. What if I use money again? If I have one quarter and I add another quarter, I have 50 cents plus 50 cents, which is 100 cents, or a dollar. So,  $1 + 1 = 2$  in terms of value. That works too. Maybe I can also think about it in terms of shapes or objects. If I have one square and another square, how many squares do I have? Two squares. So,  $1 + 1 = 2$ . That's another visual confirmation. I guess I'm overcomplicating it, but better safe than sorry. Let me try to recall if I've heard this problem before. I think it's a basic arithmetic problem that's been used in various educational settings. It's often used to introduce the concept of addition for young learners. In summary, through multiple perspectives—counting, using number lines, Roman numerals, set theory, real-world examples, and even different mathematical concepts like commutativity—it consistently points to  $1 + 1 = 2$ . I can't see any other possible answer because the fundamental operations of addition and counting lead to that result.

So, after thinking it through in so many different ways, I feel confident that  $1 + 1$  equals 2.

\*\*Final Answer\*\*: The result of  $1 + 1$  is 2. </think>

To solve the problem ( $1 + 1$ ), we can approach it from various perspectives:

1. \*\*Counting Objects\*\*: If we have one object and add another, we count them: one, two. Thus,  $(1 + 1 = 2)$ .
2. \*\*Number Lines\*\*: Starting at 1 and moving one unit to the right on the number line lands us at 2. Hence,  $(1 + 1 = 2)$ .
3. \*\*Roman Numerals\*\*: In Roman numerals,  $I (1) + I (1)$  equals  $II (2)$ .
4. \*\*Set Theory\*\*: Combining two single-element sets results in a set with two elements, so  $(1 + 1 = 2)$ .
5. \*\*Real-World Examples\*\*: Adding two dollars gives a total of two dollars, demonstrating  $(1 + 1 = 2)$ .
6. \*\*Commutative Property\*\*: Since addition is commutative,  $(1 + 1 = 1 + 1 = 2)$ .

Through multiple perspectives, the result is consistently 2. Final Answer The result of  $1 + 1$  is 2.

## Case 2: A More Complicated Question

**Query:** What is the smallest positive perfect cube that can be written as the sum of three consecutive integers?

Thinkless Response (Think Mode - 1494 Tokens)

<think>

Okay, let's try to figure out the smallest positive perfect cube that can be written as the sum of three consecutive integers. Hmm, let's start by understanding the problem step by step.

First, a perfect cube is a number raised to the power of 3. So, we need to find the smallest positive integer  $n$  such that  $n = a + (a + 1) + (a + 2)$ , where  $a$  is an integer. Then,  $n$  should also be a perfect cube. Let me write that down:

$$n = a + (a + 1) + (a + 2)$$

Simplifying the right-hand side:

$$n = 3a + 3$$

$$\text{So, } n = 3(a + 1)$$

Therefore,  $n$  must be three times some integer. But  $n$  also has to be a perfect cube. So, we need to find the smallest perfect cube that is divisible by 3.

...

I think that's thorough enough. I checked the smaller perfect cubes and confirmed that they don't work, and then verified that 27 works. So, the answer should be 27.

\*\*Final Answer\*\*

The smallest positive perfect cube that can be written as the sum of three consecutive integers is 27.

</think>

To find the smallest positive perfect cube that can be written as the sum of three consecutive integers, we start by expressing the sum of three consecutive integers. Let the three consecutive integers be  $a$ ,  $a + 1$ , and  $a + 2$ . Their sum is:

$$a + (a + 1) + (a + 2) = 3a + 3$$

This sum must be a perfect cube. Let  $n$  be the perfect cube, so we have:

$$n = 3(a + 1)$$

Thus,  $n$  must be a multiple of 3. The smallest perfect cubes are 1, 8, 27, 64, 125, etc. We check each of these to see if they can be written as  $3(a + 1)$ :

- For  $n = 1$ :  $3(a + 1) = 1$  gives  $a = -\frac{2}{3}$ , not an integer.
- For  $n = 8$ :  $3(a + 1) = 8$  gives  $a = \frac{5}{3}$ , not an integer.
- For  $n = 27$ :  $3(a + 1) = 27$  gives  $a = 8$ , which is an integer. The three consecutive integers are 8, 9, and 10, and their sum is  $8 + 9 + 10 = 27$ .

Checking the next perfect cubes (64, 125, etc.) confirms they do not yield integer values for  $a$ . Therefore, the smallest positive perfect cube that can be written as the sum of three consecutive integers is:

Okay, so I need to find the smallest positive perfect cube that can be written as the sum of three consecutive integers. Let me start by breaking down the problem.

First, let's recall that a perfect cube is a number like 1, 8, 27, 64, etc., which are cubes of integers. The question is asking for the smallest such cube that can be expressed as the sum of three consecutive numbers.

Let me represent three consecutive integers algebraically. If I let the middle integer be  $n$ , then the three consecutive numbers would be  $n-1$ ,  $n$ , and  $n+1$ . Adding these together gives:

$$(n-1) + n + (n+1) = 3n$$

So the sum of three consecutive integers is three times the middle integer. That simplifies things a bit. Therefore, the problem reduces to finding the smallest positive perfect cube that is a multiple of 3, or in other words, the cube must be divisible by 3.

...

Alternatively, if the three consecutive integers are all negative, their sum would be negative, which isn't a positive cube. Therefore, the three integers must be positive. Therefore, 27 is indeed the answer.

Hence, after all this reasoning, I think 27 is the correct answer.

\*\*Final Answer\*\*

The smallest positive perfect cube that can be written as the sum of three consecutive integers is **27**.

...

Thus, 27 is indeed the sum of three consecutive integers and is a perfect cube. Smaller cubes like 1 and 8 are not divisible by 3. Therefore, the smallest positive perfect cube that can be written as the sum of three consecutive integers is:

**27**

<think> Okay, so I need to find the smallest positive perfect cube that can be written as the sum of three consecutive integers. Hmm, let's break this down step by step.

First, let me recall what a perfect cube is. A perfect cube is a number that can be expressed as some integer raised to the power of three. So, numbers like 1 ( $1^3$ ), 8 ( $2^3$ ), 27 ( $3^3$ ), 64 ( $4^3$ ), and so on.

Next, the problem says it's the sum of three consecutive integers. Let me denote these three consecutive integers as  $n - 1$ ,  $n$ , and  $n + 1$ . So, their sum would be  $(n - 1) + n + (n + 1)$ . Let me compute that:

$$(n - 1) + n + (n + 1) = n - 1 + n + n + 1$$

Simplifying that, the  $-1$  and  $+1$  cancel out:

$$n + n + n = 3n$$

So, the sum of three consecutive integers is just three times the middle number, which is  $3n$ .

Now, we need this sum, which is  $3n$ , to be a perfect cube. So,  $3n$  must equal some integer cubed. Let me write that as an equation:

$$3n = k^3, \text{ where } k \text{ is a positive integer.}$$

So,  $n = \frac{k^3}{3}$ . For  $n$  to be an integer,  $k^3$  must be divisible by 3. Since 3 is a prime number, that means  $k$  itself must be divisible by 3. Let me denote  $k = 3m$ , where  $m$  is a positive integer. Substituting back into the equation:

$$n = \frac{(3m)^3}{3} = \frac{27m^3}{3} = 9m^3$$

So,  $n = 9m^3$ . Now, the three consecutive integers would be  $n - 1 = 9m^3 - 1$ ,  $n = 9m^3$ , and  $n + 1 = 9m^3 + 1$ . Their sum is  $3n = 3 \times 9m^3 = 27m^3$ , which is indeed a perfect cube. Now, the question is asking for the smallest positive perfect cube that can be expressed in this way. That means I need the smallest  $m$  such that  $27m^3$  is positive. Since  $m$  is a positive integer, the smallest  $m$  is 1.

Plugging  $m = 1$  into  $27m^3$ :

$$27 \times 1^3 = 27$$

So, 27 is a perfect cube and can be written as the sum of three consecutive integers. Let me verify that:

If  $m = 1$ , then  $n = 9 \times 1^3 = 9$ . So, the three consecutive integers are 8, 9, and 10.

Adding them up:  $8 + 9 + 10 = 27$ . Yep, that works.

But wait, is there a smaller cube than 27 that can be written as the sum of three consecutive integers? Let me check the next smaller cubes.

The cubes below 27 are 1, 8, and 27. 1 is too small because the sum of three consecutive integers is at least  $1 + 2 + 3 = 6$ . 8 is also too small because the next consecutive integers after 1, 2, 3 are 2, 3, 4 which sum to 9, and so on. So, 8 is  $2^3$ , but as I just thought, it's not achievable because the minimal sum is 6, which is less than 8.

Wait, hold on. Let me think about that again. If  $3n = 8$ , then  $n = 8/3$ , which is not an integer. So, 8 cannot be expressed as the sum of three consecutive integers because  $n$  must be an integer. Therefore, 8 is out.

Similarly, 1 is too small, as we saw.

So, the next cube is 27. But just to make sure, is 27 the smallest possible?

Wait, another approach: Maybe the sum of three consecutive integers doesn't have to be the same as the cube. Let me think. The sum of three consecutive integers is  $3n$ , so 27 is the cube, but is there a smaller cube that can be written as  $3n$  where  $n$  is an integer?

Looking back,  $3n$  has to be a cube. So,  $3n = k^3$ , so  $n = k^3 / 3$ . So,  $n$  must be an integer, which requires  $k^3$  divisible by 3. So,  $k$  must be divisible by 3, so  $k = 3m$ , so  $n = 9m^3$ . Thus, the smallest  $k$  is 3, giving 27. So, yes, 27 is the smallest.

Alternatively, perhaps I can think of it differently. Let me set up the equation:

Sum of three consecutive integers:  $(n - 1) + n + (n + 1) = 3n$ . So,  $3n$  is a cube. Therefore, 3 divides the cube, so 3 divides  $n$  as well. So,  $n$  must be a multiple of 3, meaning  $n = 3k$ , so  $3n = 9k$ . Therefore,  $9k$  must be a cube. So,  $9k$  is a cube, which means  $k$  must be a multiple of 3, so  $k = 3m$ , so  $9k = 27m$ , which is a cube. Therefore, the minimal  $m$  is 1, so 27 is the minimal such cube.

So, yes, 27 is the smallest positive perfect cube that can be written as the sum of three consecutive integers.

Wait a second, let me just confirm with another approach. Let me consider the sum of three consecutive integers as  $n - 1 + n + n + 1 = 3n$ . So,  $3n$  must be a cube. So,  $3n = k^3$ , so  $n = k^3 / 3$ . So,  $n$  must be integer, so  $k^3$  must be divisible by 3. Therefore,  $k$  must be a multiple of 3, so  $k = 3m$ , so  $n = (27m^3)/3 = 9m^3$ . So,  $n$  must be  $9m^3$ , which is integer for any integer  $m$ . So, the sum is  $3n = 27m^3$ , which is a cube. So, the smallest cube is when  $m=1$ , which gives 27. So, yes, 27 is the minimal.

Alternatively, maybe I can list some cubes and see if any smaller cube can be written as  $3n$ . Cubes: 1, 8, 27, 64, 125, etc.

Check if any of these can be written as  $3n$ :

1:  $3n=1 \Rightarrow n=1/3$ , not integer.

8:  $3n=8 \Rightarrow n=8/3$ , not integer.

27:  $3n=27 \Rightarrow n=9$ , which is integer.

64:  $3n=64 \Rightarrow n=64/321.333$ , not integer.

125:  $3n=125 \Rightarrow n=41.666$ , not integer.

So, indeed, 27 is the first cube where  $n$  is an integer. Therefore, 27 is the minimal.

So, the answer is 27.

\*\*Final Answer\*\* The smallest positive perfect cube that can be written as the sum of three consecutive integers is 27. </think>

To find the smallest positive perfect cube that can be written as the sum of three consecutive integers, we start by denoting these three consecutive integers as  $n - 1$ ,  $n$ , and  $n + 1$ . Their sum is:

$$(n - 1) + n + (n + 1) = 3n$$

We need this sum, which is  $3n$ , to be a perfect cube. Let  $3n = k^3$  where  $k$  is a positive integer. This implies:

$$n = \frac{k^3}{3}$$

For  $n$  to be an integer,  $k^3$  must be divisible by 3. Since 3 is a prime number,  $k$  must be divisible by 3. Let  $k = 3m$  where  $m$  is a positive integer. Substituting back, we get:

$$n = \frac{(3m)^3}{3} = 9m^3$$

Thus, the three consecutive integers are  $9m^3 - 1$ ,  $9m^3$ , and  $9m^3 + 1$ . Their sum is:

$$3 \times 9m^3 = 27m^3$$

We need the smallest positive perfect cube. The smallest  $m$  is 1, giving:

$$27 \times 1^3 = 27$$

Verifying, the three consecutive integers are 8, 9, and 10, which sum to 27. Therefore, the smallest positive perfect cube that can be written as the sum of three consecutive integers is:

27