
Think Only When You Need with Large Hybrid-Reasoning Models

Lingjie Jiang^{*†‡} Xun Wu^{*†} Shaohan Huang[†] Qingxiu Dong^{†‡} Zewen Chi[†]
Li Dong[†] Xingxing Zhang[†] Tengchao Lv[†] Lei Cui[†] Furu Wei^{†◇}
[†] Microsoft Research [‡] Peking University
<https://aka.ms/GeneralAI>

Abstract

Recent Large Reasoning Models (LRMs) have shown substantially improved reasoning capabilities over traditional Large Language Models (LLMs) by incorporating extended thinking processes prior to producing final responses. However, excessively lengthy thinking introduces substantial overhead in terms of token consumption and latency, particularly unnecessary for simple queries. In this work, we introduce Large Hybrid-Reasoning Models (LHRMs), the first kind of model capable of adaptively determining whether to perform thinking based on the contextual information of user queries. To achieve this, we propose a two-stage training pipeline comprising Hybrid Fine-Tuning (HFT) as a cold start, followed by online reinforcement learning with the proposed Hybrid Group Policy Optimization (HGPO) to implicitly learn to select the appropriate thinking mode. Furthermore, we introduce a metric called Hybrid Accuracy to quantitatively assess the model’s capability for hybrid thinking. Extensive experimental results show that LHRMs can adaptively perform hybrid thinking on queries of varying difficulty and type. It outperforms existing LRMs and LLMs in reasoning and general capabilities while significantly improving efficiency. Together, our work advocates for a reconsideration of the appropriate use of extended thinking processes, and provides a solid starting point for building hybrid thinking systems.

1 Introduction

Recent remarkable advancements in Large Reasoning Models (LRMs), such as DeepSeek-R1 [GYZ⁺25], OpenAI o1/o3 series [Ope25] and others [Tea24, MYS⁺25], have catalyzed a significant paradigm shift from conventional Large Language Models (LLMs) toward LRMs. Compared to LLMs, LRMs exhibit substantially improved reasoning capabilities and generalization across tasks such as programming, mathematics, and commonsense reasoning [XHZ⁺25, GYZ⁺25, XCM⁺25]. These improvements are primarily attributed to the generation of extended thinking traces, often marked by special tokens such as <think>, prior to producing the final responses. Despite the effectiveness of LRMs, existing work primarily focuses on converting LLMs into LRMs to enhance performance across various domains, typically through reinforcement learning algorithms such as GRPO [GYZ⁺25] and REINFORCE++ [Hu25], or by fine-tuning on distilled thinking traces [MHS⁺25].

However, broader implications regarding the potential challenges faced by LRMs in real-world applications remain underexplored. A prominent issue is the problem of **overthinking** [CXL⁺24, SCW⁺25], where excessive computational resources are expended on simple queries with negligible performance gains, even for trivial inputs such as a single word (e.g. “Hello”). These findings highlight several pressing limitations of current LRMs and raise a critical question:

^{*} Equal contribution. [◇] Corresponding author.

<p>Question 1: Let a, b, c be real numbers such that $ax^2 + bx + c \leq 1$ for all $0 \leq x \leq 1$. Find the largest possible value of $a + b + c$. Let's think step by step and output the final answer within boxed{}. Ground Truth: 17</p> <p>Qwen2.5-7B-Instruct: To find the largest possible value of $a + b + c$ given that $ax^2 + bx + c \leq 1$ for all $0 \leq x \leq 1$, we need to analyze the constraints imposed by this inequality at specific points.</p> <p>...</p> <p>All conditions are satisfied. Now, calculate $a + b + c$: $a + b + c = 2 + -2 + 1 = 2 + 2 + 1 = 5$ Thus, the largest possible value of $a + b + c$ is 5. ✗</p> <p>DeepSeek-R1-Distill-Qwen-7B: <think> Okay, so I have this problem where I need to find the largest possible value of $a + b + c$ given that $ax^2 + bx + c \leq 1$ for all x between 0 and 1. Hmm, let me think about how to approach this. ... </think></p> <p>The maximum value of $a + b + c$ is achieved when the quadratic function touches 1 at $x=0$ and $x=1$, and -1 at $x=1/2$.</p> <p>...</p> <p>For $0 \leq x \leq 1$, $0 \leq (x - \frac{1}{2})^2 \leq \frac{1}{4}$, so $-1 \leq f(x) \leq 1$. Therefore, the largest possible value of $a + b + c$ is 17 ✓</p> <p>LHRMs-7B: <think> Okay, so I need to find the largest possible value of $a + b + c$ given that the quadratic function $ax^2 + bx + c$ is bounded by 1 in absolute value for all x between 0 and 1. Hmm, let's start by understanding the problem. ... </think></p> <p>Given the problem of finding the largest possible value of $a + b + c$ under the constraint that $ax^2 + bx + c \leq 1$ for all $0 \leq x \leq 1$, we start by considering the Chebyshev polynomials and their properties.</p> <p>...</p> <p>After verifying that this polynomial indeed satisfies the given constraint and considering other potential polynomials, we conclude that the maximum value of $a + b + c$ is achieved by the polynomial $8x^2 - 8x + 1$. The largest possible value of $a + b + c$ is 17 ✓</p>	
<p>Question 2: Can you help me please?</p> <p>Qwen2.5-7B-Instruct: Of course! I'd be happy to help you. Could you please provide some more information about what you need assistance with? ✓</p> <p>DeepSeek-R1-Distill-Qwen-7B: <think> Okay, the user just said "Can you help me please?" I need to respond in a friendly and helpful way. I should acknowledge their request, show willingness to assist, and ask them to specify what they need help with. Keeping it open-ended will encourage them to provide more details.</p> <p>Hmm, maybe start with a positive affirmation. Something like, "Yes, I can help! What do you need assistance with today?" That sounds good. It's polite and inviting them to share more about their request.</p> <p>I should make sure my response is clear and concise. No need for complicated words. Just a straightforward offer to help. Let me put that together. </think></p> <p>Of course! I'm here to help. What do you need assistance with today? ✗</p> <p>LHRMs-7B: <no_think> Of course! What do you need help with? </no_think> ✓</p>	

Figure 1: Example responses from Qwen2.5-7B-Instruct, DeepSeek-R1-Distill-Qwen-7B, and our LHRMs-7B on a reasoning related task (top) and a daily QA task (bottom). While LLMs respond concisely to simple queries, but struggle with complex reasoning. LRMs handle reasoning well via explicit thinking steps but often overuse them on simple tasks, resulting in slower responses and degraded user experience. In contrast, LHRMs adaptively determines when to engage in thinking, preserving strong reasoning ability while enabling faster, more natural interactions in everyday scenarios. Additional examples are provided in Appendix E.

How to build a hybrid thinking system that can achieve an optimal balance between system 2 reasoning and system 1 ability?

To fill these gaps and make LRMs more efficient, intelligent, and better aligned with human usage requirements, we draw inspiration from human cognitive processes, where complex problems often require deliberate thinking while simple ones can be handled intuitively.

Building on this insight, we introduce **Large Hybrid-Reasoning Models (LHRMs)**, a novel class of models that, to the best of our knowledge, are the first to explicitly address the limitations of existing LRMs by dynamically deciding whether to invoke extended thinking based on the semantic and contextual characteristics of user queries. As shown in Figure 1, this adaptive thinking mechanism allows LHRMs to balance computational efficiency with task complexity, better aligning with real-world deployment scenarios and enhancing user experience without compromising performance. To achieve this, we propose a two-stage training pipeline. The first stage, Hybrid Fine-Tuning (HFT), provides an effective cold start by enabling the model to learn to support two distinct thinking modes on the same query without mode collapse. This is followed by Hybrid Group Policy Optimization, a variant of online RL designed to teach the model when to engage in extended thinking while simultaneously generating more helpful and harmless responses. Furthermore, we introduce a metric termed Hybrid Accuracy, which correlates strongly with human expert judgment, to quantitatively evaluate the model’s capability for hybrid thinking.

Extensive experimental results and human studies conducted on Qwen-2.5 series models ranging from 1.5B to 7B parameters across multiple domains (including mathematics, programming, and general tasks) demonstrate that our LHRMs effectively performs hybrid thinking by adapting to queries of varying difficulty and types. Moreover, LHRMs outperforms existing LRMs and LLMs

in both reasoning and general capabilities, while significantly improving efficiency. In general, our main contribution is as follows:

1. We introduce Large Hybrid-Reasoning Models, the first kind of model which can adaptively determine whether to perform thinking based on the contextual information of user queries, mimicking human reasoning patterns and mitigating the over-thinking problem.
2. We develop a two-stage training pipeline comprising Hybrid Fine-Tuning as a cold start and a novel RL method, Hybrid Group Policy Optimization, which enables the model to learn when to engage in thinking while simultaneously generating more helpful and harmless responses.
3. Extensive experiments across diverse model scales and domains demonstrate that LHRMs accurately performs hybrid thinking and significantly improves both efficiency and general capabilities compared to existing LRMs and LLMs.

2 Large Hybrid-Reasoning Models

2.1 Problem Formulation

Specifically, let q be an input query. The LHRMs has access to two thinking modes: *Thinking* and *No-Thinking*, denoted as $\mathcal{M} = \{\top, \bot\}$, respectively. Each mode defines a inducing a conditional distribution $\mathcal{P}(a \mid q, m)$ over answers $a \in \mathcal{A}$, i.e.,

$$\mathcal{P}(a \mid q, m), \quad m \in \mathcal{M}. \quad (1)$$

For each query q , the model aims to select the optimal reasoning mode $m^*(q)$ from the candidate set \mathcal{M} , such that a task-specific utility function $\mathcal{U}(q, a)$ is maximized in expectation:

$$m^*(q) = \arg \max_{m \in \mathcal{M}} \mathbb{E}_{a \sim \mathcal{P}(a \mid q, m)} [\mathcal{U}(q, a)]. \quad (2)$$

The overall objective is to learn a policy $\pi : \mathcal{P} \rightarrow \mathcal{M}$ to select a mode $m \in \mathcal{M}$ that maximizes the expected utility $\mathcal{U}(q, A)$ over the query distributions $\Theta = \{(\mathcal{D}_i, \mathcal{U}_i)\}_{i=1}^N$:

$$\max_{\pi} \frac{1}{N} \sum_{i=1}^N \mathbb{E}_{\mathcal{D}_i \sim \Theta, \mathcal{D}_i \leftrightarrow \mathcal{U}_i} \left[\mathbb{E}_{a \sim \mathcal{P}(a \mid q, \pi(q)), q \sim \mathcal{D}_i} [\mathcal{U}_i(q, a)] \right]. \quad (3)$$

We summarize two key challenges and our corresponding solutions as follow:

C1. How to learn the policy π based on the \mathcal{U}_i . We first propose a two-stage training phase to learn a effective LHRMs by: Hybrid Fine-Tuning as code start (§ 2.2), following by online reinforcement learning with Hybrid Group Policy Optimization (§ 2.3) to guides LHRMs to more effectively select between two reasoning modes to generate more appropriate and high-quality responses, respectively.

C2. How to evaluate the hybrid thinking capability of a policy π . We propose a evaluation metric named hybrid accuracy, which is designed to assess the hybrid thinking ability of models in § 2.4.

2.2 Stage I: Hybrid Fine-Tuning

We begin with a hybrid-formatted supervised fine-tuning stage named Hybrid Fine-Tuning (HFT) that integrates both reasoning-intensive (*Thinking*) and direct-answer (*No-Thinking*) data. This approach mitigates the instability often observed in cold-start scenarios [GYZ⁺25], and establishes a robust initialization for next stage reinforcement learning.

Data Construction. The hybrid fine-tuning dataset consists of both reasoning-intensive and direct-answer examples. The think-style subset includes high-quality math, code, and science questions sourced from existing datasets [MJB⁺25, Fac25, PLK⁺25, Tea25, XLY⁺25], with answers generated by Deepseek-R1 [GYZ⁺25] and verified for correctness. Each instance is annotated with `<think>` and `</think>` tags to mark intermediate reasoning steps. For the non-think-style subset, we collect simple queries from WildChat-1M [ZRH⁺24] using a FastText-based classifier [JGB⁺16] to exclude complex reasoning tasks. The remaining factual and conversational examples are wrapped with `<no_think>` and `</no_think>` tags to indicate that direct answers are sufficient.

After deduplication and the removal of overlaps with evaluation benchmarks, we obtain a final set of 1.7M hybrid-formatted training examples. This curated collection provides a high-quality,

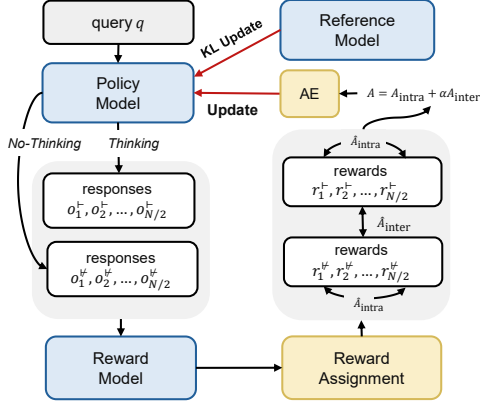


Figure 2: Demonstration of Hybrid Group Policy Optimization. HGPO proceeds by: (1) sampling multiple responses for each query q using both reasoning modes; (2) scoring the responses with the reward model and assigning these rewards based on Eq. 9; and (3) computing the advantage and policy loss, followed by updating the policy model. AE denotes advantage estimator and reward assignment denotes Eq. 9.

Algorithm 1 Hybrid Group Policy Optimization

Input model trained at Stage I $\pi_{\theta_{\text{HFT}}}$; reward models R_ϕ ; queries \mathcal{P} ; hyperparameters ϵ, β, μ

```

1: policy model  $\pi_\theta \leftarrow \pi_{\theta_{\text{HFT}}}$ 
2: for iteration = 1, ..., I do
3:   reference model  $\pi_{\text{ref}} \leftarrow \pi_\theta$ 
4:   for step = 1, ..., M do
5:     Sample a batch  $\mathcal{P}_b$  from  $\mathcal{P}$ 
6:     Update the old policy model  $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$ 
7:     Sample  $N$  outputs  $\mathcal{O} \sim \pi_{\theta_{\text{old}}}(\cdot | q, m)$  for
        $q \in \mathcal{P}_b$  and  $m \in \mathcal{M}$ 
8:     Compute rewards  $r(o_i^m)$  for  $o_i^m \in \mathcal{O}$  by
       running  $R_\phi$ 
9:     Assign each  $r(o_i^m)$  to  $r_{\text{inter}}(o_i^m)$  and
        $r_{\text{intra}}(o_i^m)$  using Eq. 9
10:    Compute  $A_i^t$  for the  $t$ -th token of  $o_i^m$ 
       through Eq. 10
11:    for iteration = 1, ...,  $\mu$  do
12:      Update the  $\pi_\theta$  by maximizing the objec-
       tive  $\mathcal{J}_{\text{HGPO}}$  in Eq. 11
13:    end for
14:  end for
15: end for
Output  $\pi_\theta$ 

```

format-consistent foundation for cold-start training, enabling the model to effectively handle both reasoning-heavy and direct-response tasks.

Optimize Objective. HFT trains the model to predict the next token based on prior context. For the constructed dataset $\mathcal{D}_{\text{HFT}} = \{(x^i, y^i)\}_{i=1}^N$, the objective is:

$$\mathcal{L}_{\text{HFT}}(\theta) = -\mathbb{E}_{[(x,y) \sim \mathcal{D}_{\text{HFT}}]} \left[\sum_{t=1}^{|y|} \log \pi_\theta(y_t | x, y_{1:t-1}) \right], \quad (4)$$

2.3 Stage II: Hybrid Group Policy Optimization

After Stage I, LHRMs acquires an initial ability to support two distinct reasoning modes in \mathcal{M} on the same query q without collapsing. However, our ultimate goal is to enable the LHRMs to adaptively select the most appropriate reasoning mode $m^*(q)$ (Eq. 2) for each query q to improve its applicability in real-world scenarios. To this end, we propose a effective RL algorithm named Hybrid Group Policy Optimization (HGPO) to explicitly teach the model how to adaptively select between reasoning modes, i.e., to learn the policy π in Eq. 3, while simultaneously improving the model’s foundational ability (e.g., helpfulness and harmlessness).

HGPO is illustrated in Figure 2 and Algorithm 1. Following prior RL approaches [ACG⁺24, Hu25, SWZ⁺24], HGPO discards the critic model and instead estimates the values using multiple samples for each prompt q to reduce the computational cost training. Specifically, for each question q in the task prompt set \mathcal{P} , HGPO samples two groups of outputs \mathcal{O} from the old policy $\pi_{\theta_{\text{HFT}}}$, using *Thinking* and *No-Thinking* mode, respectively.

Sampling Strategy. Given a query $q \in \mathcal{P}$, we sample N candidate responses from the old policy $\pi_{\theta_{\text{HFT}}}$ under two distinct reasoning modes $\mathcal{M} = \{\vdash, \not\vdash\}$. Specifically,

$$\{o_i^{\vdash}\}_{i=1}^{N/2} \sim \pi_{\theta_{\text{HFT}}}(\cdot | q, m = \vdash), \quad \{o_i^{\not\vdash}\}_{i=1}^{N/2} \sim \pi_{\theta_{\text{HFT}}}(\cdot | q, m = \not\vdash) \quad (5)$$

We define the complete output candidate set as:

$$\mathcal{O}(q) = \{o_i^{\vdash}\}_{i=1}^{N/2} \cup \{o_i^{\not\vdash}\}_{i=1}^{N/2}. \quad (6)$$

Reward Scoring and Assignment. A reward function R_ϕ ² is applied to each candidate output, yielding two reward sets:

$$\mathcal{R}^\vdash = \{r(o_i^\vdash)\}_{i=1}^{N/2}, \quad \mathcal{R}^\nabla = \{r(o_i^\nabla)\}_{i=1}^{N/2}. \quad (7)$$

Since the reward scores assigned by the reward model may vary significantly across different domains, we apply a rule-based assignment scheme to normalize the existing reward scores. Specifically, we compute two types of binary reward: **inter-group rewards** r_{inter} and **intra-group rewards** r_{intra} to jointly capture both the relative quality across different reasoning modes and the answer quality within each individual reasoning mode. Given the average reward for each mode computed as

$$\bar{\mathcal{R}}^\vdash = \frac{2}{N} \sum_{i=1}^{N/2} r(o_i^\vdash), \quad \bar{\mathcal{R}}^\nabla = \frac{2}{N} \sum_{i=1}^{N/2} r(o_i^\nabla), \quad (8)$$

we have r_{inter} and r_{intra} assigned as:

$$r_{\text{inter}}(o_i^m) = \begin{cases} 1, & \text{if } m = \arg \max_{m' \in \{\vdash, \nabla\}} \{\bar{\mathcal{R}}^\vdash, \bar{\mathcal{R}}^\nabla + \delta\} \\ 0, & \text{otherwise} \end{cases}, \quad r_{\text{intra}}(o_i^m) = \begin{cases} 1, & \text{if } i = \arg \max_{j \in \{1, \dots, N/2\}} r_j^m \\ 0, & \text{otherwise} \end{cases}. \quad (9)$$

Here margin δ is a hyperparameter that governs the trade-off between two reasoning modes.

Advantage Estimation. After getting the final two kinds of rewards for \mathcal{O} , we can adopt advantage estimators like REINFORCE++ [Hu25], GRPO [SWZ⁺24], and RLOO [ACG⁺24] to estimate the advantages. Here we use GRPO as the default advantage estimator, i.e.,

$$A_i^t = \underbrace{\left[\frac{r_{\text{intra}}(o_i) - \text{mean}(r_{\text{intra}}(o_j))}{\text{std}(r_{\text{intra}}(o_j))} \right]}_{\text{GRPO for intra-group advantage } A_{\text{intra}}} + \underbrace{\mathbb{1}\{o_i^t \in \Phi\} \cdot \alpha \left[\frac{r_{\text{inter}}(o_i) - \text{mean}(r_{\text{inter}}(o_j))}{\text{std}(r_{\text{inter}}(o_j))} \right]}_{\text{GRPO for inter-group advantage } A_{\text{inter}}}, \quad (10)$$

where A_i^t is the final estimated per-token advantage for each response in \mathcal{O} , $\Phi = \{\text{<think>, <no_think>}\}$ and α is the hyperparameter for controlling the weight of inter-group advantages. We also explore the impact of using different estimators in § 3.3.

Optimize Objective. Following [ACG⁺24, Hu25, SWZ⁺24], HGPO optimizes the policy model π_θ by maximizing the following objective:

$$\mathcal{J}_{\text{HGPO}}(\theta) = \mathbb{E}_{[q \sim \mathcal{P}, \{o_i^m\}_{i=1}^N \sim \pi_{\theta_{\text{HFT}}}(\mathcal{O}|q), m \in \mathcal{M}]} \left[\frac{1}{N} \sum_{i=1}^N \sum_{t=1}^{|o_i|} \left[\min \left(\frac{\pi_\theta(o_i^{m,t}|q, o_i^{m,<t})}{\pi_{\theta_{\text{HFT}}}(o_i^{m,t}|q, o_i^{m,<t})} A_i^t, \text{clip} \left(\frac{\pi_\theta(o_i^{m,t}|q, o_i^{m,<t})}{\pi_{\theta_{\text{HFT}}}(o_i^{m,t}|q, o_i^{m,<t})}, 1 - \epsilon, 1 + \epsilon \right) A_i^t \right) - \beta \mathbb{D}_{\text{KL}}(\pi_\theta || \pi_{\text{ref}}) \right], \quad (11)$$

where

$$\mathbb{D}_{\text{KL}}(\pi_\theta || \pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(o_i^m|q)}{\pi_\theta(o_i^m|q)} - \log \frac{\pi_{\text{ref}}(o_i^m|q)}{\pi_\theta(o_i^m|q)} - 1. \quad (12)$$

Here $\pi_{\theta_{\text{HFT}}}$ denotes the model we get from Stage I in § 2.2 and ϵ and β are hyper-parameters.

2.4 Evaluating Hybrid Thinking Capability

To more comprehensively evaluate the performance of LHRMs, beyond conventional downstream task metrics, we propose a new metric called **Hybrid Accuracy** (\mathcal{H}_{Acc}), which aims to measure the LHRMs' ability to correctly select the appropriate reasoning pattern.

Given the task prompt set $\mathcal{P} = \{p_i\}_{i=1}^K$, LHRMs is first applied to sample N responses under two distinct reasoning modes \vdash and ∇ . A reward model is then employed as a scorer to evaluate and assign scores to both sets of generated responses. The mode with the higher average score is regarded as the ground-truth preferred reasoning mode for each prompt, denoted as m_{gt} . In cases where the average scores of the two modes are equal or the difference between them falls below a predefined margin, the mode yielding the shorter response is selected as m_{gt} . Subsequently, we allow LHRMs to autonomously select a reasoning mode m_p for each prompt. The proportion of prompts for which LHRMs' selected mode matches the ground-truth mode is reported as

$$\mathcal{H}_{\text{acc}} = \frac{1}{K} \sum_{i=1}^K \mathbb{1}[\text{Equal}(m_{\text{gt}}, m_p)] \quad \text{s.t.} \quad m_{\text{gt}}, m_p \in \{\vdash, \nabla\}. \quad (13)$$

²For queries with definitive answers, we use rule-based rewards [SWZ⁺24, GYZ⁺25] for a better reward estimation; otherwise, a trained parametric reward model is applied.

3 Experimental Results

3.1 Experimental Setup

Compared Baselines. To validate the effectiveness of our proposed method LHRMs, we conduct a comprehensive comparison against state-of-the-art LLMs and LRMs derived from the same base models. Specifically, we build our LHRMs on both 1.5B and 7B parameter versions of Qwen-2.5-math-base [YZH⁺24], and compare our method with several Qwen-2.5-derived variants, including:

- **LLMs:** we compare our model with Qwen-2.5 Math series [YZH⁺24] and Instruct series [YYZ⁺24], which show great capabilities in coding, mathematics and general tasks.
- **LRMs:** Here we use DeepSeek-R1-Distill series [DA25], which are distilled using the reasoning data generated by DeepSeek-R1 [DA25] and attain strong reasoning ability.
- **Hybrid:** Due to the absence of existing models capable of hybrid thinking, we compare our final model against the baseline we obtained in our Stage I (§ 2.2) (denoted as HFT). Additionally, we construct two further baselines by applying Direct Preference Optimization (DPO) [RSM⁺23] and Rejection Sampling Fine-Tuning (RFT) [YYL⁺23] to the checkpoint $\pi_{\theta_{\text{HFT}}}$ from Stage I, using the same training data as in Stage II (§ 2.3). These baselines are referred to as HFT-DPO and HFT-RFT, respectively. Implementation details of DPO and RFT can be found in Appendix A.

Evaluation Settings. We primarily evaluate model performance based on the following aspects:

- **Reasoning Capabilities.** We evaluate models on a comprehensive set of widely-used reasoning benchmarks, including math related like MATH500 [LKB⁺23], AIME24 [AM24], AMC23, Olympiad Bench [HLB⁺24], and code related like LiveCodeBench [JHG⁺24], MBPP [AON⁺21] and MBPP+ [LXW⁺24].
- **General Capabilities.** We assess the models’ general capabilities through open-ended generation tasks using LLMs as judges. Specifically, we adopt AlpacaEval 2.0 [DGLH24] and Arena-Hard [LCF⁺24] to assess instruction-following ability and alignment with human preferences.
- **Hybrid Thinking Capabilities.** We compute Hybrid Accuracy (\mathcal{H}_{Acc}) (§ 2.4) on MATH500 to evaluate the model’s ability to correctly select the appropriate reasoning mode.

More details about the evaluation settings can be found in Appendix D.

Training Settings. For stage I, we finally obtained 1.7 M SFT data and details about construction pipeline, sources and statistics can be found in § 2.2 and Appendix C. All models are trained for 3 epochs with the AdamW optimizer, employing a 10% linear warmup followed by a cosine learning rate decay schedule. The maximum learning rate is set to $1\text{e}-4$, with a batch size of 128 and a maximum sequence length of 32k tokens. Training the 7B model in the SFT phase takes approximately 2.5 days on 4 nodes of NVIDIA 8×H100 stations.

For stage II, we construct the training dataset by randomly sampling 76K queries from DeepScaler [LTW⁺25] and Tulu3 [LMP⁺24] (details can be found in Appendix C). We use Llama-3.1-Tulu-3-8B-RM³ as the parametric reward model in Eq. 7. We use VeRL [SZY⁺24] to conduct experiments. By default, we use a constant 1×10^{-6} learning rate together with AdamW optimizer for policy model, and use a batch size of 256 and micro batchsize of 8. The rollout stage collects 256 prompts and samples 4 responses for each prompt. We set $\alpha = 1.0$ and margin = 0.2 for RL training. We set KL coefficient to 0.001 and $\epsilon = 0.5$ in Eq. 11 in all experiments. The RL phase takes 2 days on NVIDIA 4×H100 Stations.

3.2 Main Results

Overall Performance. The overall results of different models on the aforementioned benchmarks are presented in Table 1. We observe that LHRMs consistently outperforms all comparable baselines across both the 1.5B and 7B model scales, We observe that LHRMs consistently outperforms all comparable baselines across both the 1.5B and 7B model scales, achieving average improvements of **9.2%** and **7.1%** compared to HFT-DPO, and **18.3%** and **11.5%** compared to HFT-RFT, at the 1.5B and 7B scales, respectively. Specifically, it surpasses the strongest competing baseline, HFT-DPO, by **10.3%** and **13.6%** on the AIME24 benchmark at 1.5B and 7B scales, respectively. On the 7B

³<https://huggingface.co/allenai/Llama-3.1-Tulu-3-8B-RM>

Table 1: Performance comparison across different tasks. “-” indicates that the model does not support hybrid thinking and is therefore left blank. The last column (Avg.) reports the average performance across all evaluation tasks, excluding the \mathcal{H}_{acc} metric. Bold numbers indicate the best performance. *Type* refers to the model’s reasoning mode, where *Hybrid* denotes models that adaptively select between *Thinking* and *No-Thinking* modes for each query.

Methods	Type	MATH				Code			General		\mathcal{H}_{acc}	Avg.
		MATH500	AIME24	AMC23	Olympiad	LiveCode	MBPP	MBPP+	Alpaca	Arena		
1.5B size model												
Qwen2.5-Math-1.5B	LLMs	42.4	3.3	22.5	16.7	0.4	16.1	14.3	0.1	1.8	-	13.1
Qwen2.5-1.5B-Instruct	LLMs	51.0	3.3	52.8	38.7	2.2	60.1	51.9	8.8	1.1	-	30.0
Qwen2.5-Math-1.5B-Instruct	LLMs	72.0	6.7	60.0	38.1	3.7	26.7	23.8	2.8	4.7	-	26.5
DeepSeek-R1-Distill-Qwen-1.5B	LRMs	83.9	28.9	62.9	43.3	16.8	54.2	46.3	5.6	2.7	-	38.3
HFT-1.5B	Hybrid	87.8	32.7	75.0	48.9	15.7	54.8	47.4	13.1	6.9	41.4	42.5
HFT-RFT-1.5B	Hybrid	82.2	22.0	67.5	44.1	14.2	49.7	42.6	13.6	8.5	48.1	38.3
HFT-DPO-1.5B	Hybrid	86.8	32.6	75.0	48.7	17.2	50.5	42.6	13.3	6.9	45.8	41.5
LHRMs-1.5B	Hybrid	87.8	35.3	75.0	50.4	17.2	61.1	54.0	16.9	10.4	54.4	45.3
7B size model												
Qwen2.5-Math-7B	LLMs	57.0	13.3	22.5	21.8	6.0	31.5	27.3	2.0	7.0	-	20.9
Qwen2.5-7B-Instruct	LLMs	77.0	13.3	52.8	29.1	14.6	79.9	67.5	36.2	25.8	-	44.0
Qwen2.5-Math-7B-Instruct	LLMs	82.4	10.0	62.5	41.6	2.6	40.2	34.7	3.8	10.0	-	32.0
DeepSeek-R1-Distill-Qwen-7B	LRMs	92.8	55.5	91.5	58.1	37.6	74.3	64.3	19.1	17.9	-	56.8
HFT-7B	Hybrid	93.6	56.7	95.0	58.5	34.7	70.6	59.8	23.7	14.0	34.2	56.4
HFT-RFT-7B	Hybrid	87.8	55.3	82.5	55.0	35.8	81.0	68.8	28.1	14.0	49.7	56.6
HFT-DPO-7B	Hybrid	93.8	58.7	92.5	60.6	38.8	80.1	68.3	23.3	13.0	37.1	58.9
LHRMs-7B	Hybrid	93.8	66.7	95.0	61.2	38.8	81.5	69.6	35.0	26.0	71.9	63.1

scale, LHRMs further outperforms HFT-DPO by **50.2%** on Alpaca and **93.4%** on Arena-Hard. Notably, AIME24 and Arena-Hard represent the most challenging benchmarks in the math and general domains, respectively. These results demonstrate the strong reasoning and general capability of our LHRMs.

Furthermore, we find that our LHRMs achieves the best hybrid thinking performance, as measured by \mathcal{H}_{acc} , significantly outperforming all baselines. For example, it exceeds HFT-DPO by **93.8%** and RFT by **44.7%**. This further demonstrates that our HGPO effectively enables the model to learn correct hybrid thinking behaviors, providing a promising and practical pathway for building hybrid thinking systems. Next, we provide a detailed investigation into each of the two training stages.

Effect of HFT at Stage I. By comparing HFT with the Qwen2.5 and Deepseek-R1-Distill series, we observe that HFT significantly enhances both reasoning and general capabilities, while demonstrating robust cold-start performance by maintaining stability during hybrid reasoning without failure or collapse. These results validate the effectiveness of the proposed HFT (§ 2.2) framework.

Effect of HGPO at Stage II. By comparing LHRMs with HFT, we find that HGPO further substantially improves the model’s reasoning and general capabilities, while enabling more effective selection of thinking modes (over **31.4%** and **110.2%** \mathcal{H}_{acc} improvements at 1.5B and 7B size, respectively). These findings demonstrate the effectiveness of HGPO.

Furthermore, when comparing LHRMs with HFT-DPO and HFT-RFT, we observe that LHRMs achieves superior downstream performance on both reasoning and general tasks, along with higher accuracy in reasoning mode selection. This highlights the effectiveness of HGPO.

Notably, we also observe that LHRMs exhibits stronger cross-domain generalization capabilities. Although RL training is conducted solely on math and general-domain data, LHRMs-1.5B achieves substantial improvements on code-related tasks, with gains of **11.1%** on MBPP and **13.9%** on MBPP+. In contrast, both HFT-DPO-1.5B and HFT-RFT-1.5B show experience performance drops (**7.8%** and **9.3%**, respectively) on the same tasks. This indicates that LHRMs is able to generalize learned hybrid thinking patterns across domains—a property not observed in DPO or RFT.

3.3 Analysis

Effect of Different Advantage Estimators. To investigate the impact of different advantage estimators on HGPO training, we replace GRPO in Eq. 10 with RLOO [ACG⁺24] and REIN-

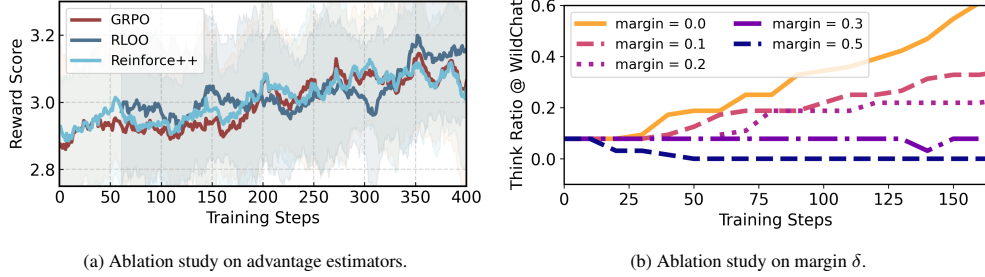


Figure 3: Ablation study on the effects of advantage estimators and margin δ . δ in Eq. 9.

FORCE++ [Hu25]. Implementation details can be found in Appendix B. As shown in Figure 3 (a), all estimators yield competitive performance, indicating that HGPO is robust to the choice of advantage estimator.

Effect of Margin δ in Eq. 9. We investigate how varying the margin δ influences the model’s bias toward the two reasoning modes. As shown in Figure 3 (b), different values of δ result in distinct hybrid reasoning behaviors. Specifically, a larger δ encourages the model to favor the *No-Thinking* mode. This suggests that δ can serve as a control knob for tailoring hybrid reasoning behavior to specific application needs. For instance, a higher δ may be preferred when real-time responsiveness is prioritized, whereas a lower δ is more suitable when reasoning quality is the primary concern.

Thinking Ratio Study within Domain. Figure 4 illustrates the distribution of thinking ratio across varying difficulty levels in the MATH500 benchmark for both HFT-7B and LHRMs-7B. We observe that HFT-7B maintains a consistently high thinking ratio across all difficulty levels. In contrast, after applying the Stage II HGPO, LHRMs-7B exhibits a decreasing thinking ratio as the difficulty level decreases while getting a higher overall performance (see in Table 1). This trend suggests that our HGPO can effectively enables the model to adaptively perform hybrid thinking based on the input query. Specifically, the model learns to respond to simpler queries using a *No-Thinking* strategy, thereby reducing reasoning cost and inference latency without sacrificing accuracy. For more challenging queries, the model consistently engages in full thinking, achieving higher reasoning accuracy where it matters most.

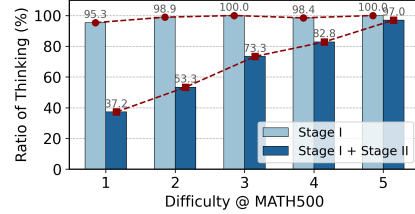


Figure 4: Analysis of thinking ratio of LHRMs within a single domain.

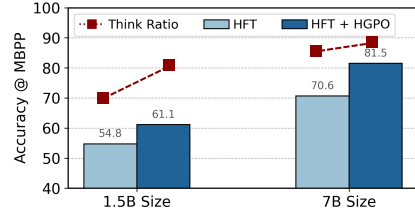


Figure 5: Analysis of thinking ratio of LHRMs across different domains.

Specifically, the model learns to respond to simpler queries using a *No-Thinking* strategy, thereby reducing reasoning cost and inference latency without sacrificing accuracy. For more challenging queries, the model consistently engages in full thinking, achieving higher reasoning accuracy where it matters most.

Thinking Ratio Study across Domain. We investigate how HGPO affects model performance on unseen domains not encountered during the RL phase. Specifically, we conduct RL using data from the math and general domains, and then evaluate the resulting models on the code domain. As shown in Figure 5, we observe that the model attained from HFT exhibits a relatively low think ratio in the code domain. However, as RL progresses—despite being conducted on unrelated domains such as math—the think ratio in the code domain gradually increases, accompanied by improved performance. This suggests that our HGPO is capable of transferring hybrid thinking patterns learned in a specific domain (e.g., math) to other domains (e.g., code), demonstrating strong generalization and transferability.

Thinking Ratio Study across Model Size. We analyze how model size influences hybrid thinking behavior during the RL phase. In Figure 6, we present the think ratio of the 1.5B and 7B models

Question: If $a * b = 9a + 2b - ab + 5$, what is the value of $5 * 1$?

DeepSeek-R1-Distill-Qwen-7B: <think> Okay, let's tackle this problem. So, we have this operation defined as a star b equals 9a plus 2b minus ab plus 5. And we need to find the value of 5 star 1. Hmm, let me make sure I understand the question... (2941 characters) </think> To solve $5 * 1$ using the defined operation $a * b = 9a + 2b - ab + 5$:

1. Substitute $a = 5$ and $b = 1$ into the expression: $5 * 1 = 9(5) + 2(1) - (5)(1) + 5$
2. Calculate each term: $- 9 \times 5 = 45 - 2 \times 1 = 2 - 5 \times 1 = 5$
3. Substitute these values back into the expression: $45 + 2 - 5 + 5$
4. Perform the arithmetic operations step-by-step: $- 45 + 2 = 47 - 47 - 5 = 42 - 42 + 5 = 47$

Thus, the value of $5 * 1$ is 47 ✓

LHRMs-7B: <no_think> To solve the problem, we need to substitute $a = 5$ and $b = 1$ into the given function $a * b = 9a + 2b - ab + 5$. Step-by-step solution:

1. Start with the given function: $a * b = 9a + 2b - ab + 5$
2. Substitute $a = 5$ and $b = 1$ into the function: $5 * 1 = 9(5) + 2(1) - 5(1) + 5$
3. Perform the multiplication operations: $5 * 1 = 45 + 2 - 5 + 5$
4. Simplify the expression by performing the addition and subtraction operations: $5 * 1 = 45 + 2 - 5 + 5 = 47$

Therefore, the final answer is: 47 ✓</no_think>

Figure 7: Example outputs from DeepSeek-R1-Distill-Qwen-7B and our LHRMs-7B model. For easy questions, LHRMs selects the *No-Thinking* mode.

on the MATH500 benchmark. We observe that, as RL training progresses, the think ratio of the 7B model gradually decreases, while that of the 1.5B model increases. This indicates that hybrid thinking behavior during RL is strongly correlated with model capacity and knowledge storage. Specifically, larger models with stronger knowledge reserves tend to shift towards using *No-Thinking* strategies on solvable problems to reduce reasoning overhead and improve response efficiency. In contrast, smaller models are more inclined to invoke *Thinking* more frequently, as a way to compensate for their limited capabilities through enhanced reasoning.

Case Analysis. We present representative model responses in Figure 1 and 7. We observe that for relatively simple queries—such as everyday questions and basic math questions, LHRMs tends to autonomously select the *No-Thinking* mode, achieving fast and accurate responses. In contrast, for more complex problems that require deeper reasoning, LHRMs adaptively switches to the *Thinking* mode to produce more precise solutions. More example responses can be found in Appendix E.

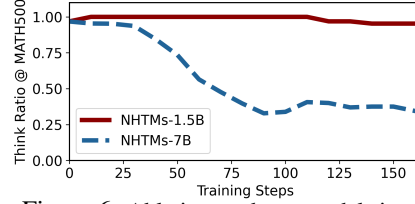


Figure 6: Ablation study on model size.

4 Related Works

Test-Time Scaling (TTS). Test-time scaling [Fac25, GYZ⁺25, MYS⁺25, IBR⁺23] has been validated as a promising approach to enhance model performance beyond scaling model size [SLXK24]. There are two primary approaches to implementing TTS [MYS⁺25]: **Parallel** and **Sequential**. The former is achieved by sampling multiple solutions and select the one by a selector (e.g., an outcome reward model), like Best-of-N [BJE⁺24, IBR⁺23, Lev24], and Monte-Carlo Tree Search (MCTS) [LCP⁺24, ZCS⁺23, ZYSR⁺24, CFWS23]. The latter aims to achieve TTS by enabling the model to generate longer outputs, i.e., Chain-of-Thoughts (CoT) within a single sampling pass by prompting, finetuning or reinforcement learning. Beyond the field of NLP, TTS has also been shown to effectively improve the test-time performance of trained models in other domains, like image generation, video generations [LWC⁺25] and multi-modality learning [FGL⁺25, HJZ⁺25, CLZ⁺25].

Large Reasoning Models. Recent advances in Large Reasoning Models (LRMs), such as DeepSeek-R1 [DA25], OpenAI o1/o3 series [Ope25], and others [Tea24, MYS⁺25, Ant25, Goo25], have led to a growing focus on Large Reasoning Models (LRMs). Compared to general LLMs, LRMs extend the capabilities of LLMs by generating long chains of reasoning steps with reflection before outputting the final answers. LRMs are typically developed by applying reinforcement learning such as GRPO [GYZ⁺25] and REINFORCE++ [Hu25], or distilled from stronger models [Tea25, MJB⁺25, Fac25, ZWP⁺25, YHX⁺25].

5 Conclusion

In this work, we focus on building a large language model that effectively balances reasoning capabilities and general-purpose performance. To this end, we propose a novel evaluation metric, \mathcal{H}_{acc} , designed to consistently assess a model’s ability to perform hybrid thinking across diverse tasks. We further introduce a two-stage training pipeline consisting of HFT and HGPO. Experimental results demonstrate that this pipeline significantly improves performance on both reasoning-centric and general downstream tasks. Moreover, it enhances the model’s hybrid thinking capabilities, leading to a better user experience by reducing unnecessary reasoning on simple queries commonly observed in LRMs, and mitigating the insufficient reasoning capabilities in conventional LLMs.

References

- [ACG⁺24] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to Basics: Revisiting REINFORCE-Style Optimization for Learning from Human Feedback in LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2024, Bangkok, Thailand, August 11-16, 2024, pages 12248–12267. Association for Computational Linguistics, 2024.
- [AM24] AI-MO. Aime 2024, 2024.
- [Ant25] Anthropic. Claude 3.7 sonnet and claude code. <https://www.anthropic.com/news/claude-3-7-sonnet>, 2025.
- [AON⁺21] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
- [BJE⁺24] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024.
- [CFWS23] Sehyun Choi, Tianqing Fang, Zhaowei Wang, and Yangqiu Song. Kcts: Knowledge-constrained tree search decoding with token-level hallucination detection, 2023.
- [CLZ⁺25] Liang Chen, Lei Li, Haozhe Zhao, Yifan Song, and Vinci. R1-v: Reinforcing super generalization ability in vision-language models with less than \$3. <https://github.com/Deep-Agent/R1-V>, 2025. Accessed: 2025-02-02.
- [CXL⁺24] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.
- [DA25] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
- [DGLH24] Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaEval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.
- [Fac25] Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025.
- [FGL⁺25] Kaituo Feng, Kaixiong Gong, Bohao Li, Zonghao Guo, Yibing Wang, Tianshuo Peng, Benyou Wang, and Xiangyu Yue. Video-r1: Reinforcing video reasoning in mllms. *arXiv preprint arXiv:2503.21776*, 2025.
- [Goo25] Google. Gemini 2.5 flash. <https://deepmind.google/technologies/gemini/flash/>, 2025.
- [GYZ⁺25] 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.
- [HJZ⁺25] Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.
- [HLB⁺24] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024.

- [Hu25] Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. arXiv preprint arXiv:2501.03262, 2025.
- [IBR⁺23] Robert Irvine, Douglas Boubert, Vyas Raina, Adian Liusie, Ziyi Zhu, Vineet Mudupalli, Aliaksei Korshuk, Zongyi Liu, Fritz Cremer, Valentin Assassi, Christie-Carol Beauchamp, Xiaoding Lu, Thomas Rialan, and William Beauchamp. Rewarding chatbots for real-world engagement with millions of users, 2023.
- [JGB⁺16] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, H  rve J  gou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. arXiv preprint arXiv:1612.03651, 2016.
- [JHG⁺24] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. arXiv preprint arXiv:2403.07974, 2024.
- [KKVR⁺23] Andreas K  pf, Yannic Kilcher, Dimitri Von R  tte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Rich  rd Nagyfi, et al. Openassistant conversations-democratizing large language model alignment. Advances in Neural Information Processing Systems, 36:47669–47681, 2023.
- [LCF⁺24] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. arXiv preprint arXiv:2406.11939, 2024.
- [LCP⁺24] Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Don’t throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding, 2024.
- [Lev24] Noam Levi. A simple model of inference scaling laws, 2024.
- [LKB⁺23] 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.
- [LMP⁺24] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. T  lu 3: Pushing frontiers in open language model post-training, 2024.
- [LTW⁺25] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.
- [LWC⁺25] Fangfu Liu, Hanyang Wang, Yimo Cai, Kaiyan Zhang, Xiaohang Zhan, and Yueqi Duan. Video-t1: Test-time scaling for video generation. arXiv preprint arXiv:2503.18942, 2025.
- [LXW⁺24] Jiawei Liu, Songrun Xie, Junhao Wang, Yuxiang Wei, Yifeng Ding, and Lingming Zhang. Evaluating language models for efficient code generation. arXiv preprint arXiv:2408.06450, 2024.
- [LXWZ23] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. Advances in Neural Information Processing Systems, 36:21558–21572, 2023.

- [MHS⁺25] Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset. arXiv preprint arXiv:2504.16891, 2025.
- [MJB⁺25] Justus Mattern, Sami Jaghouar, Manveer Basra, Jannik Straube, Matthew Di Ferrante, Felix Gabriel, Jack Min Ong, Vincent Weisser, and Johannes Hagemann. Synthetic-1: Two million collaboratively generated reasoning traces from deepseek-r1, 2025.
- [MLL⁺25] Sadegh Mahdavi, Muchen Li, Kaiwen Liu, Christos Thrampoulidis, Leonid Sigal, and Renjie Liao. Leveraging online olympiad-level math problems for llms training and contamination-resistant evaluation. arXiv preprint arXiv:2501.14275, 2025.
- [MYS⁺25] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. arXiv preprint arXiv:2501.19393, 2025.
- [Ope25] OpenAI. Openai gpt-4.5 system card. OpenAI Publication, 2025.
- [PLK⁺25] Guilherme Penedo, Anton Lozhkov, Hynek Kydlíček, Loubna Ben Allal, Edward Beeching, Agustín Piqueres Lajarín, Quentin Gallouédec, Nathan Habib, Lewis Tunstall, and Leandro von Werra. Codeforces. <https://huggingface.co/datasets/open-r1/codeforces>, 2025.
- [RSM⁺23] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. Advances in Neural Information Processing Systems, 36:53728–53741, 2023.
- [SCW⁺25] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, et al. Stop overthinking: A survey on efficient reasoning for large language models. arXiv preprint arXiv:2503.16419, 2025.
- [SLXK24] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. arXiv preprint arXiv:2408.03314, 2024.
- [SWZ⁺24] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300, 2024.
- [SZY⁺24] 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.
- [Tea24] Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown, November 2024.
- [Tea25] OpenThoughts Team. Open Thoughts. <https://open-thoughts.ai>, January 2025.
- [WWS⁺22] 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.
- [XCM⁺25] Zhihui Xie, Liyu Chen, Weichao Mao, Jingjing Xu, Lingpeng Kong, et al. Teaching language models to critique via reinforcement learning. arXiv preprint arXiv:2502.03492, 2025.
- [XHZ⁺25] F Xu, Q Hao, Z Zong, J Wang, Y Zhang, J Wang, X Lan, J Gong, T Ouyang, F Meng, et al. Towards large reasoning models: A survey of reinforced reasoning with large language models. arxiv. 10.48550. arXiv preprint arXiv.2501.09686, 2025.

- [XLY⁺25] Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. Kodcode: A diverse, challenging, and verifiable synthetic dataset for coding, 2025.
- [YHX⁺25] Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning, 2025.
- [YYL⁺23] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. [arXiv preprint arXiv:2308.01825](#), 2023.
- [YYZ⁺24] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. [arXiv preprint arXiv:2412.15115](#), 2024.
- [YZH⁺24] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. [arXiv preprint arXiv:2409.12122](#), 2024.
- [ZCS⁺23] Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B. Tenenbaum, and Chuang Gan. Planning with large language models for code generation, 2023.
- [ZRH⁺24] Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatGPT interaction logs in the wild. In [The Twelfth International Conference on Learning Representations](#), 2024.
- [ZWP⁺25] Han Zhao, Haotian Wang, Yiping Peng, Sitong Zhao, Xiaoyu Tian, Shuaiting Chen, Yunjie Ji, and Xiangang Li. 1.4 million open-source distilled reasoning dataset to empower large language model training, 2025.
- [ZYSR⁺24] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models, 2024.
- [ZZZ⁺24] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyang Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In [Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics \(Volume 3: System Demonstrations\)](#), Bangkok, Thailand, 2024. Association for Computational Linguistics.

Contents

1	Introduction	1
2	Large Hybrid-Reasoning Models	3
2.1	Problem Formulation	3
2.2	Stage I: Hybrid Fine-Tuning	3
2.3	Stage II: Hybrid Group Policy Optimization	4
2.4	Evaluating Hybrid Thinking Capability	5
3	Experimental Results	6
3.1	Experimental Setup	6
3.2	Main Results	6
3.3	Analysis	7
4	Related Works	9
5	Conclusion	10
A	Implement Details for DPO and RFT	16
B	Implement Details for RLOO and Reinforce++	16
C	Dataset Statistics	17
D	Evaluation Settings	17
E	Example Outputs	18

A Implement Details for DPO and RFT

In this section, we detail the construction pipeline of raining data for both DPO and RFT used in § 3. To construct datasets for DPO and RFT, we adopt the same set of queries used in HGPO for offline sampling. Specifically, for each query q , we sample two responses using each of the two thinking modes $m \in \{\top, \bot\}$, resulting in a set of four responses per query as

$$\mathcal{O}(q) = \{o_i^\top\}_{i=1}^2 \cup \{o_i^\bot\}_{i=1}^2. \quad (14)$$

Then the reward function R_ϕ^4 in Eq. 15 is applied to score these responses as:

$$\mathcal{R}^\top = \{r(o_i^\top)\}_{i=1}^2, \quad \mathcal{R}^\bot = \{r(o_i^\bot)\}_{i=1}^2. \quad (15)$$

Given the average reward for each mode computed as

$$\bar{\mathcal{R}}^\top = \frac{1}{2} \sum_{i=1}^2 r(o_i^\top), \quad \bar{\mathcal{R}}^\bot = \frac{1}{2} \sum_{i=1}^2 r(o_i^\bot), \quad (16)$$

DPO. For DPO, the training dataset which contains win and lose sample is constructed as following:

$$\mathcal{D}_{\text{DPO}} = \left\{ (q, o_w, o_l) \mid o_w = \arg \max_{o \in \{o_i^{\top w}\}_{i=1}^2} (r(o)), \quad o_l = \arg \min_{o \in \{o_i^{\top l}\}_{i=1}^2} (r(o)) \right\}, \quad (17)$$

while

$$m_w = \arg \max_{m \in \{\top, \bot\}} \bar{\mathcal{R}}^m, \quad m_l = \arg \min_{m \in \{\top, \bot\}} \bar{\mathcal{R}}^m. \quad (18)$$

After getting \mathcal{D}_{DPO} , we optimize the model π_θ initialized from $\pi_{\theta_{\text{HFT}}}$ by using training objective:

$$\max_{\pi_\theta} \mathbb{E}_{(x, o_w, o_l) \sim \mathcal{D}_{\text{DPO}}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(o_w | x)}{\pi_{\theta_{\text{HFT}}}(o_w | x)} - \beta \log \frac{\pi_\theta(o_l | x)}{\pi_{\theta_{\text{HFT}}}(o_l | x)} \right) \right]. \quad (19)$$

RFT. For RFT, the training dataset is constructed as following:

$$\mathcal{D}_{\text{RFT}} = \left\{ (q, o) \mid o = \arg \max_{o \in \mathcal{O}} (\mathcal{R}^\top \cup \mathcal{R}^\bot) \right\}. \quad (20)$$

The training objectives for RFT are presented as:

$$\mathcal{L}_{\text{RFT}}(\theta) = -\mathbb{E}_{[(x, o) \sim \mathcal{D}_{\text{RFT}}]} \left[\sum_{t=1}^{|o|} \log \pi_\theta(o_t | x, o_{1:t-1}) \right], \quad (21)$$

For implementations, we use LLaMA-Factory [ZZZ⁺24]⁵ as the codebase for both DPO and RFT. To ensure a fair comparison, we maintain the same learning rate, batch size, and total number of training samples as used in Stage II of HGPO.

B Implement Details for RLOO and Reinforce++

First of all, We compare different advantage estimators including REINFORCE++ [Hu25], GRPO [SWZ⁺24], and RLOO [ACG⁺24], toggling the existence of our HGPO. To make different algorithms compatible with the compound of intra-group rewards and inter-group rewards, we accordingly make adaption similar to Eq. 10. For Reinforce++, we have

$$A_i^t = \underbrace{\sum_{s=t}^{|o_i|} \gamma^{s-t} \cdot r_{\text{intra}}(o_i)}_{\text{REINFORCE++ for intra-group advantage } A_{\text{intra}}} + \underbrace{\mathbb{1}\{o_i^t \in \Phi\} \cdot \alpha [r_{\text{inter}}(o_i)]}_{\text{REINFORCE++ for inter-group advantage } A_{\text{inter}}}, \quad (22)$$

⁴Same with § 2.3, for queries with definitive answers, we use rule-based rewards [SWZ⁺24, GYZ⁺25]; otherwise, a trained parametric reward model [LMP⁺24] is applied.

⁵<https://github.com/hiyouga/LLaMA-Factory.git>

Here is a hyperparameter representing the decay factor, which is set to 0.99 in our experiments. For RLOO, we have

$$A_i^t = \underbrace{\left[r_{\text{intra}}(o_i) - \frac{1}{N-1} \sum_{j \neq i} r_{\text{intra}}(o_j) \right]}_{\text{RLOO for intra-group advantage } A_{\text{intra}}} + \mathbb{1}\{o_i^t \in \Phi\} \cdot \alpha \underbrace{\left[r_{\text{inter}}(o_i) - \frac{1}{N-1} \sum_{j \neq i} (r_{\text{inter}}(o_j)) \right]}_{\text{RLOO for inter-group advantage } A_{\text{inter}}}. \quad (23)$$

From § 3.3, We show that **HGPO is a general plug-in for almost any advantage estimators.**, which largely extends the use cases of HGPO. We implement both RLOO and Reinforce++ on VeRL [SZY⁺24]⁶.

C Dataset Statistics

Stage I. Figure 8 shows the token length distributions for the *Thinking* and *No-Thinking* datasets we construct in Stage I. The *Thinking* data has an average length of 575 tokens, with the 25th and 75th percentiles at 362 and 672 tokens, respectively, and a maximum length of 9,148 tokens. In contrast, the *No-Thinking* data exhibits a significantly higher average length of 4,897 tokens, with the 25th and 75th percentiles at 1,719 and 6,738 tokens, and a maximum of 23,997 tokens.

We present the scores and statistics of our constructed dataset for HFT in Table 2. The dataset covers a diverse range of domains, primarily including reasoning-intensive tasks such as mathematics and code, as well as general-purpose question answering.

Stage II. We report the details of training data used in Stage II at Table 3.

Table 2: Data distribution and source of Stage I.

Category	Source	Data Size	Total	Link
General	WildChat-1M [ZRH ⁺ 24]	649,569	674,908	https://huggingface.co/datasets/allenai/WildChat-1M
	OSSAT2 [KKVR ⁺ 23]	25,339		https://huggingface.co/datasets/OpenAssistant/oasst1
MATH	SYNTHETIC-1 [MJB ⁺ 25]	343,988	631,325	https://huggingface.co/datasets/PrimeIntellect/SYNTHETIC-1
	OpenR1-Math [Fac25]	93,533		https://huggingface.co/datasets/open-r1/OpenR1-Math-220k
	OpenThought [Tea25]	55,566		https://huggingface.co/datasets/open-thoughts/OpenThoughts-114k
	AoPS [MLL ⁺ 25]	137,497		https://huggingface.co/datasets/di-zhang-fdu/AOPS
	AIME	741		https://huggingface.co/datasets/di-zhang-fdu/AIME_1983_2024
Coding	SYNTHETIC-1 [MJB ⁺ 25]	107,543	381,845	https://huggingface.co/datasets/PrimeIntellect/SYNTHETIC-1
	OpenR1-Codeforces [PLK ⁺ 25]	8,926		https://huggingface.co/datasets/open-r1/codeforces
	OpenThought [Tea25]	19,447		https://huggingface.co/datasets/open-thoughts/OpenThoughts-114k
	KodCode [XLY ⁺ 25]	245,929		https://huggingface.co/datasets/KodCode/KodCode-V1
Others	SYNTHETIC-1 [MJB ⁺ 25]	6,508	6,508	https://huggingface.co/datasets/PrimeIntellect/SYNTHETIC-1
Total	—	1,694,586	1,694,586	—

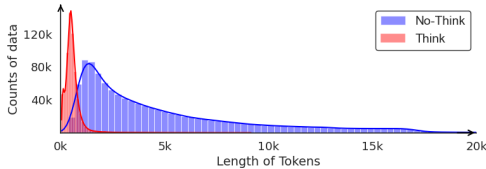


Figure 8: Token length distributions of *Thinking* and *No-Thinking* data in Stage I.

Table 3: Data distribution and source of Stage II. Note that our experiments are conducted on subsets of the two datasets rather than their full versions.

Category	Source	Data Size	Used
General	Tulu 3 [LMP ⁺ 24]	337,000	50,000
	Tulu 3 preference dataset		
MATH	DeepScaler [LTW ⁺ 25]	40,300	40,000
	DeepScaleR-Preview-Dataset		

D Evaluation Settings

In this section, we introduce the details for downstream tasks evaluation in § 3. During inference, we set the temperature to 0.6, the top-p value to 0.95, and the maximum generation length to 35,000 tokens.

⁶<https://github.com/volcengine/verl.git>

- For MATH500, AIME24, GPQA Diamond, and LiveCodeBench, we adopt the OpenR1 [Fac25] evaluation framework. For HumanEval+ and MBPP+, we use the EvalPlus [LXWZ23, LXW⁺24] framework with Chain-of-Thought (CoT) [WWS⁺22] prompting. We report pass@1 for all reasoning related benchmarks.
- For AlpacaEval 2.0 and Arena-Hard, we use GPT-4-Turbo and GPT-4-0314 as the respective baseline models, with GPT-4o serving as the judge in both evaluations. To reduce length-related bias, we feed only the final generated summary into the evaluation in think mode.

We provide a summarization and links of evaluation tools we used in experiments in Table 4.

Table 4: Summarization of evaluation tools we used in experiments.

Task	Tool Name	Source
MATH500 [LKB ⁺ 23], AIME24 [AM24], Livecodebench [JHG ⁺ 24]	open-r1 [Fac25]	https://github.com/huggingface/open-r1
Olympiad Bench, AMC23	math-evaluation-harness	https://github.com/ZubinGou/math-evaluation-harness
MBPP [AON ⁺ 21, LXW ⁺ 24], MBPP+ [AON ⁺ 21, LXW ⁺ 24]	evalplus [LXWZ23, LXW ⁺ 24]	https://github.com/evalplus/evalplus
AlpacaEval [DGLH24]	aplaca_eval	https://github.com/tatsu-lab/alpaca_eval
ArenaHard v2.0 [LCF ⁺ 24]	arena-hard-auto	https://github.com/lmarena/arena-hard-auto

E Example Outputs

Question: What is the value of $513^2 - 487^2$? Let's think step by step and output the final answer within boxed{}. Ground Truth: 26000	
DeepSeek-R1-Distill-Qwen-7B: <think> Okay, so I need to calculate 513 squared minus 487 squared. Hmm, let me think about how to approach this. I remember that there's a formula for the difference of squares, which is $a^2 - b^2 = (a - b)(a + b)$. Maybe I can use that here instead of calculating each square separately, which might be time-consuming. ... (3837 characters) </think>	
To solve $513^2 - 487^2$, we use the difference of squares formula $a^2 - b^2 = (a - b)(a + b)$.	
1. Identify $a = 513$ and $b = 487$. 2. Calculate $a - b$:	
	$513 - 487 = 26$
3. Calculate $a + b$:	
	$513 + 487 = 1000$
4. Multiply the results from steps 2 and 3:	
	$26 \times 1000 = 26000$
Verification by direct calculation: $513^2 = 263169$ - $487^2 = 237169$ - Subtracting these values: $263169 - 237169 = 26000$	
Both methods confirm the result.	
Therefore, the final answer is 26000 ✓	
LHRMs-7B: <no_think> To solve $513^2 - 487^2$, we can use the difference of squares formula, which states that $a^2 - b^2 = (a + b)(a - b)$. Here, $a = 513$ and $b = 487$.	
Step 1: Calculate $a + b$:	
	$513 + 487 = 1000$
Step 2: Calculate $a - b$:	
	$513 - 487 = 26$
Step 3: Use the difference of squares formula:	
	$513^2 - 487^2 = (513 + 487)(513 - 487)$
	$513^2 - 487^2 = 1000 \times 26$
Step 4: Perform the multiplication:	
	$1000 \times 26 = 26000$
Therefore, the final answer is: 26000 </no_think> ✓	

Figure 9: LHRMs-7B uses the No-Thinking mode to solve easy math problems.

Question: You are given an array of integers `nums` of length `n` and a positive integer `k`. The power of an array is defined as: its maximum element if all of its elements are consecutive and sorted in ascending order, and `-1` otherwise. You need to find the power of all subarrays of `nums` of size `k`. Return an integer array `results` of size `n - k + 1`, where `results[i]` is the power of `nums[i..(i + k - 1)]`.

HFT-7B:

<no_think> To solve this problem, we need to evaluate the power of all subarrays of a given size `k` from the input array `nums`. The power of a subarray is defined as its maximum element if all elements are consecutive and sorted in ascending order; otherwise, it is `-1`.

Here's a step-by-step plan to solve the problem:

1. **Iterate through each subarray of size `k`:** We will slide a window of size `k` through the array `nums` and examine each subarray.
2. **Check if the subarray is sorted and consists of consecutive integers:**
 - First, check if the subarray is sorted in ascending order.
 - Then, check if the elements of the subarray are consecutive integers.
3. **Calculate the power of each subarray:** If the subarray meets the criteria, the power is the maximum element; otherwise, it is `-1`.

Here is the Python code that implements this logic:

```
class Solution:
    def resultsArray(self, nums: List[int], k: int) -> List[int]:
        n = len(nums)
        result = []

        for i in range(n - k + 1):
            subarray = nums[i:i + k]
            sorted_subarray = sorted(subarray)
            max_element = subarray[-1]

            if subarray == sorted_subarray and max_element - subarray[0] == k - 1:
                result.append(max_element)
            else:
                result.append(-1)

        return result
```

</no_think> ✗

LHRMs-7B:

<think> Okay, I need to solve this problem where I have to find the power of all subarrays of size `k` in the given array. The power is defined as the maximum element if all elements are consecutive and sorted in ascending order. Otherwise, it's `-1`. Hmm, let's break down the problem. So for each subarray of length `k`, I have to check two things: whether the elements are consecutive and whether they are sorted in ascending order. ... (2645 characters) </think>

To solve this problem, we need to determine the power of all subarrays of a given size `k` from an array of integers. The power of a subarray is defined as the maximum element if the subarray is both sorted in ascending order and consists of consecutive integers. Otherwise, the power is `-1`.

Approach

1. **Iterate through each possible subarray:** For each starting index `i` from 0 to '`n - k`', extract the subarray of length `k`.
2. **Check if the subarray is sorted:** Verify if each element in the subarray is less than the next element. If not, the power is `-1`.
3. **Check if the subarray consists of consecutive integers:** Ensure each element is exactly one more than the previous element. If not, the power is `-1`.
4. **Determine the power:** If both conditions are met, the power is the last element of the subarray (since it is the maximum in a sorted consecutive sequence). Otherwise, the power is `-1`.

```
class Solution:
    def resultsArray(self, nums: List[int], k: int) -> List[int]:
        result = []
        n = len(nums)
        for i in range(n - k + 1):
            sub = nums[i:i+k]
            # Check if the subarray is sorted in ascending order
            is_sorted = True
            for j in range(len(sub) - 1):
                if sub[j] >= sub[j+1]:
                    is_sorted = False
                    break
            if not is_sorted:
                result.append(-1)
                continue
            # Check if the subarray consists of consecutive integers
            is_consecutive = True
            for j in range(len(sub) - 1):
                if sub[j+1] != sub[j] + 1:
                    is_consecutive = False
                    break
            if is_consecutive:
                result.append(sub[-1])
            else:
                result.append(-1)
        return result
```

✓

Figure 10: LHRMs-7B learned to select the Thinking mode to solve hard code problems.