From Chatbots to Reasoners – Reasoning in Generative AI

Junwei Huang Chenyue Fang

ABSTRACT

The paradigm for advanced Generative AI is shifting from probabilistic pattern-matching to deliberate reasoning. This report provides a comprehensive analysis of the core technologies enabling this transition and investigates the architectural innovations of state-of-the-art models. A key contribution is an empirical attempt to replicate the DeepSeek-R1 training pipeline on a smaller-scale model. Our experimental results reveal a critical trade-off in base model selection during post-training, underscoring the significant hurdles in democratizing advanced reasoning alignment techniques. The report concludes with a summary of current limitations and future prospects for the development of reasoning in AI.

1 Introduction

OpenAI once described their definition of 5 levels of AI: Chatbots, Reasoners, Agents, Creators, and ultimately, fully integrated organizations. The first stage includes non-reasoning LLM's such as OpenAI's GPT-4o. While their capabilities are already remarkable, their operational paradigm is fundamentally based on probabilistic pattern matching, "predicting the next token" based on learned distributions from vast training corpora. This approach, though effective, seems not to be what people truly expect as "Intelligence".

The critical next frontier is therefore the transition from probabilistic chatbots to true "reasoners": models that can engage in logical thought, construct coherent argumentative chains, and solve complex problems from first principles. Recent emergence of reasoning models including OpenAI-o1 and DeepSeek-R1 marks a pivotal moment, demonstrating the feasibility of incentivizing and embedding robust reasoning capabilities within LLMs. These models signify a departure from pure pattern replication towards a more profound form of machine intelligence.

This report aims to dissect the core technologies underpinning this breakthrough. We seek to move beyond a surface-level performance analysis to investigate the fundamental algorithms and training pipelines—such as advanced reinforcement learning and preference optimization techniques—that enable these models to reason. Furthermore, a primary objective of our work is to explore the feasibility of replicating these sophisticated methods on smaller, more accessible models through local deployment and fine-tuning experiments.

To achieve these objectives, this report is organized as follows: Section 2 provides an overview of the core technologies central to modern reasoning models. Section 3 examines several archetypal reasoning models, with a focus on OpenAI o1 and DeepSeek-R1. Section 4 presents the methodology and results of our experimental work, detailing the local deployment and fine-tuning of a smaller model. A detailed analysis of our experimental results and their implications is also included. Finally, Section 5 concludes with a brief summary of the limitations and future directions of reasoning in Generative AI.

2 Methodology

Chain-of-Thought

Early since 2022 (far before the emergence of reasoning LLMs), researchers have been interested in discovering the internal reasoning ability of language models. Few-shot Chain-of-Thought (CoT) prompting[1] is one of the earliest and perhaps most illustrative technique behind this idea. Apart from simple input-output examples, prompt engineers provide the language model with a few demonstrations that explicitly lay out the intermediate reasoning steps to reach a final answer. The model learns to replicate this reasoning pattern on new, complex tasks. This approach yields significant improvements in performance and shows human-readable rationales, making the model's conclusions more transparent and interpretable. However, this method relies on manual prompt engineering, which is time-consuming and task-specific, limiting its implementation to general tasks.

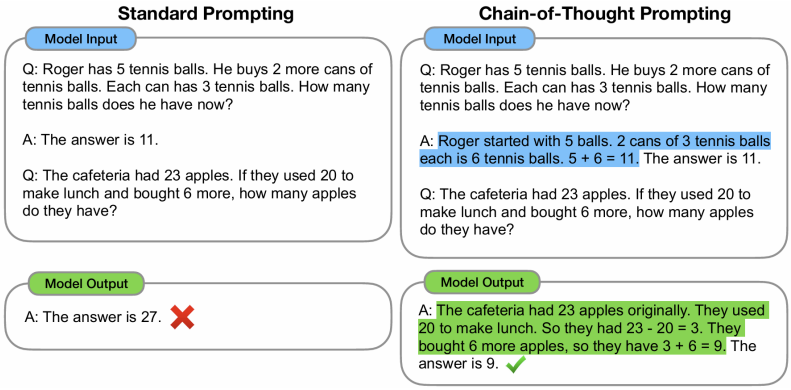


Figure 1. In-context learning of Few-shot CoT

To overcome these challenges, Zero-shot CoT Prompting[2] was introduced. By adding a magic spell "Let's think step by step", the method successfully leverages the inherent reasoning capabilities of LLMs, triggering the model to break down its reasoning process without any prior examples. This makes the power of CoT accessible to a much broader audience and applicable across a wide array of tasks. The method also demonstrated the "scaling law" between model size and it's intrinsic reasoning capability.

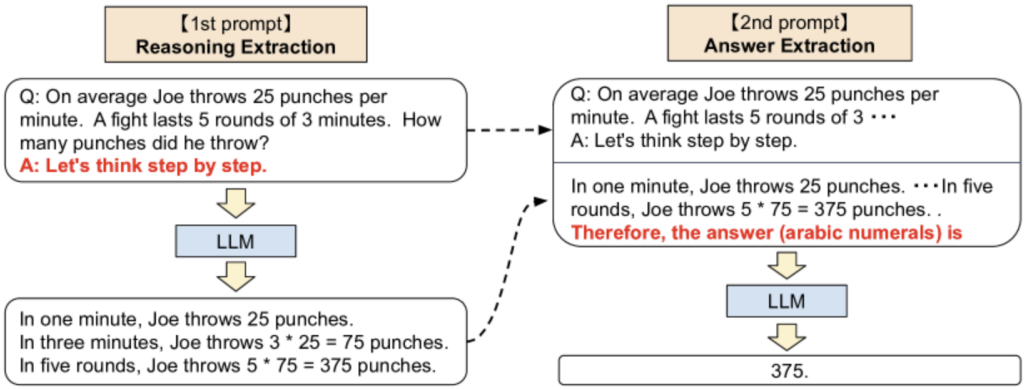


Figure 2. Example of Zero-shot CoT

Building on the core principles of CoT, more sophisticated techniques have been developed to further refine and stabilize the reasoning process. Self-Consistency, for example, improves robustness by generating multiple reasoning paths for a single problem and then selecting the most consistent answer through a majority vote. This mitigates the risk of a single, flawed reasoning chain leading to an incorrect result.

A more dynamic approach is the Tree of Thoughts (ToT) framework[3]. This method allows an LLM to explore multiple reasoning paths simultaneously, structuring them as a tree. The model is able to "look ahead" to evaluate potential paths, and "backtrack" if a certain reasoning path proves unpromising. This process of self-evaluation and deliberate decision-making allows the model to tackle more complex problems that require exploration and correction, mimicking a more human-like problem-solving process.

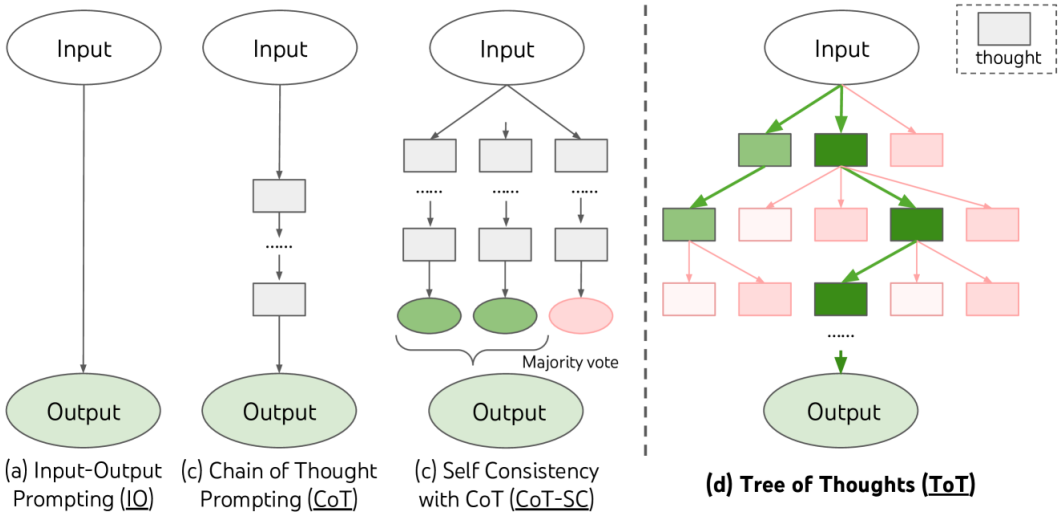


Figure 3. Illustration of Tree of Thoughts

Self-improving prompts

With the continuous growth of LLM's capability, a natural idea occurred: Can LLM generate better prompts than human? Inspired by this concept of Self-improving Prompts, Automated Prompt Engineering[4] is introduced. Instead of relying on human intuition, this approach cleverly uses LLMs themselves to discover the most effective instructions for any given task.

The methodology is centered around a sophisticated Iterative Optimization loop, including:

1. **Prompts Generation:** An inference model generates a diverse set of candidate prompts, ranging from simple instructions like "write the antonym of the word" to more complex variations.

2. **Prompts Evaluation:** Each generated prompt is passed to a scoring model. The model then conduct the task using the specific prompt, and the score is evaluated by measuring the log probability of producing the correct answer.

3. **Iterative Monte Carlo Search:** A search algorithm is used to navigate the vast space of possible prompts, retaining high-scoring candidates while discarding those that perform poorly. This cycle of generating, scoring, and selecting is repeated, while the system optionally resamples and creates variations of the best prompts to further refine them.

APE substituted static, human-designed prompts by dynamic, machine-honed instructions that unlock a higher level of performance from LLMs. As an example, the famous prompt “Let’s think step by step” is optimized to a more explicit version, “Let’s work this out in a step by step way to be sure we have the right answer”. The idea of letting LLM to improve themselves also inspired many future works.

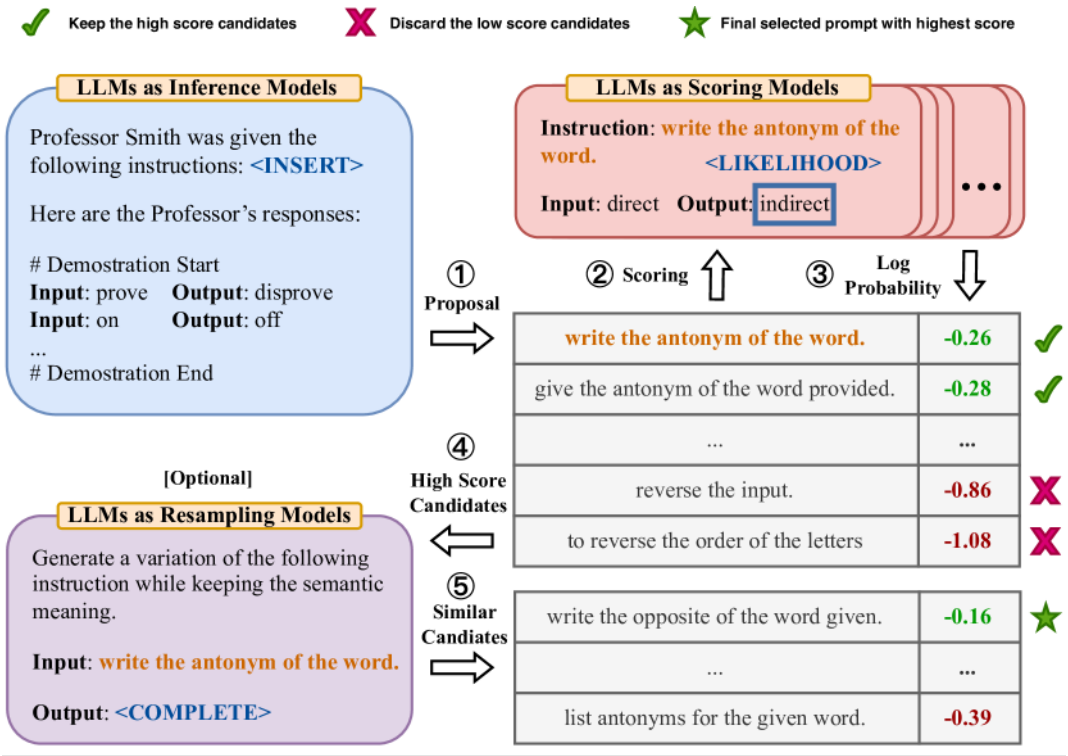


Figure 4. Iterative Process of APE

Reinforcement Learning with Human Feedback[5]

Reinforcement Learning from Human Feedback (RLHF) is an advanced hybrid training paradigm designed to align the behavior of Large Language Models (LLMs) with human preferences and values. This technique integrates supervised learning with reinforcement learning through a three-phase process. The core idea is to train a "reward model" that serves as a scalable proxy for human preferences, and then use this model to guide the optimization of a policy model(see Figure 5).

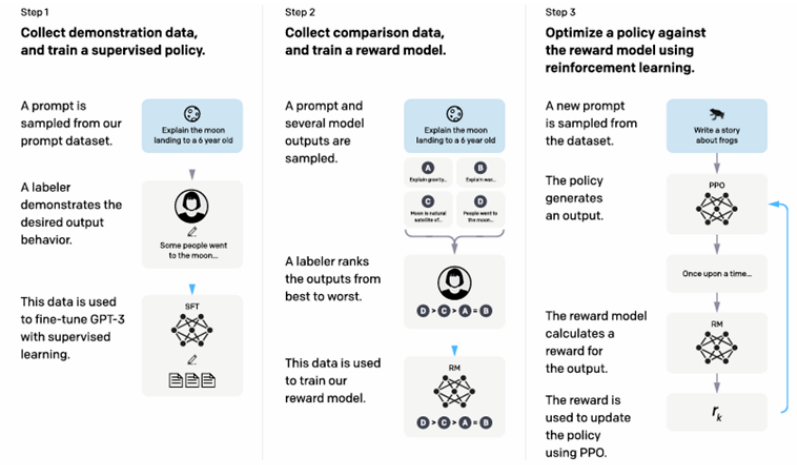


Figure 5. A diagram illustrating the three steps of our method: (1) supervised fine-tuning (SFT), (2) reward model (RM) training, and (3) reinforcement learning via proximal policy optimization (PPO) on this reward model.

The RLHF Training Pipeline

Phase 1: Supervised Fine-Tuning (SFT)

Objective: To establish a high-quality initial policy by adapting a pre-trained model to the instruction-following domain.

Process:

1. **Data Collection:** A diverse set of prompts is first collected or curated from a broad corpus.
2. **Human Demonstration:** Human labelers demonstrate the desired responses for each prompt.
3. **Supervised Training:** The pre-trained language model is then fine-tuned on this dataset of prompt-demonstration pairs using standard supervised learning methods.

Outcome: A Supervised Fine-Tuned (SFT) model. This model possesses the foundational ability to follow instructions, but may lack optimal consistency and reliability. It serves as the basis for data generation in Phase 2 and policy optimization in Phase 3.

Phase 2: Reward Model (RM) Training

Objective: To train a model that can evaluate the quality of a model's output and return a reward score. The purpose of this Reward Model (RM) is to learn and quantify human preferences for what constitutes a "good" versus a "bad" response.

Process:

1. **Data Generation:** For a given prompt, the SFT model from Phase 1 is used to generate multiple distinct outputs.

2. **Human Ranking of Comparison Data:** These outputs are presented to human labelers who rank them from best to worst based on a set of criteria (e.g. relevance, factuality, harmlessness).

3. **Reward Model Training:** These ranked data are used to train the reward model. Its input is a prompt-response pair, and its output is a scalar reward value.

Outcome: A trained Reward Model. This model functions as an automated and scalable proxy for human judgment, providing critical feedback for reinforcement learning in Phase 3.

Phase 3: Reinforcement Learning via Proximal Policy Optimization (PPO)

Objective: To further fine-tune the the SFT model using feedback from the reward model, aligning its responses with human preferences.

Process:

1. **Policy Initialization:** The SFT model from Phase 1 is used as the initial policy model (π^{SFT}).

2. **RL Loop:**

    a. A prompt x is sampled from the dataset.

    b. The current policy model, π_ϕ^{RL}, generates a response y given the prompt x.

    c. The trained Reward Model, r_θ, from Phase 2 evaluates the response y and provides a scalar reward, $r_\theta(x,y)$.

    d. The parameters of the policy model are updated using the reward signal via the adapted Proximal Policy Optimization (PPO-ptx) algorithm with following the objective function:

$$\begin{aligned}\text{objective}(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\text{RL}}(y \mid x)/\pi^{\text{SFT}}(y \mid x)\right)\right] \\ +\gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\text{RL}}(x))\right]\end{aligned}$$

Outcome: A final, aligned language model that has undergone the full RLHF pipeline. This model demonstrates superior performance in following complex instructions and generating human-preferred responses.

Summary and Broader Implications Essentially, RLHF is an alignment process that seeks to align the model's behavior with human preferences. However, this alignment procedure comes at the cost of lower performance on certain tasks, which known as the "alignment tax". To mitigate this, the PPO-ptx algorithm incorporates a pre-training mix-in penalty term into the PPO objective function, as shown in the equation above. This term ensures that improvements in alignment do not come at a significant cost to its fundamental performance.

To show that alignment research is an important topic in reasoning AI, several broader lessons can be drawn :

1. **Cost-Effectiveness of Alignment:** The computational cost of alignment via RLHF is a small fraction of that required for pre-training. Yet, it still effectively makes language models more helpful to users.

2. **Generalization of Instruction-Following:** The aligned model demonstrates an ability to generalize its instruction-following capabilities to tasks and domains on which it was not explicitly supervised. This is a crucial property, as it is infeasible to supervise a model on all possible tasks.

3. **Low Alignment Tax:** The methodology successfully mitigated significant performance degradations on common benchmarks, demonstrating that RLHF can be a "low-tax" alignment technique. A low alignment tax is critical for encouraging the adoption of safety and alignment measures in future highly capable AI systems.

Proximal Policy Optimization[6]

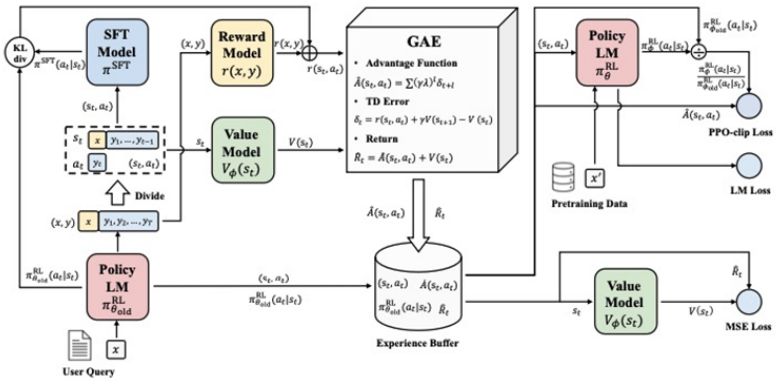


Figure 6. An overview of the PPO-based RLHF training pipeline.

Proximal Policy Optimization (PPO) is an on-policy, model-free reinforcement learning algorithm designed to achieve stable and efficient policy updates. The training process is iterative and can be broken down into the following key stages (See Figure 6):

1. **Policy Rollout and Experience Collection** The current policy, denoted as π_θ, interacts with the environment to sample a batch of trajectories. Each trajectory is a sequence of states, actions, and rewards (s_t, a_t, r_t) collected over a finite time horizon. This on-policy data directly reflects the performance of the current policy.

2. **Advantage Function Estimation using GAE** For each time step t in the collected trajectories, the advantage function $\hat{A}_t$ is calculated. This is typically achieved using the Generalized Advantage Estimation (GAE) algorithm. The advantage $\hat{A}_t$ quantifies how much better a given action a_t is compared to the average action at state s_t, effectively reducing the variance of the policy gradient estimate. This calculation requires a value function estimate $V(s_t)$, which is often learned concurrently by a separate critic network.

3. **Policy Optimization via the PPO Objective Function** The parameters θ of the policy network are updated by optimizing the PPO objective function L using the collected data. PPO primarily utilizes

a clipped surrogate objective function to ensure stable updates. The core idea is to constrain the magnitude of the policy change at each iteration.

- **Clipped Surrogate Objective (L^{CLIP}):** This is the most common PPO variant. It defines an objective that "clips" the probability ratio between the new and old policies. This clipping disincentivizes large policy updates that could destabilize the training process, thereby ensuring more monotonic improvements.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t) \right]$$

  Where ε is a hyperparameter, generally set to be 0.1 or 0.2, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the ratio of current policy and old policy, $\hat{A}_t$ is the advantage estimate at time step t.

- **Adaptive KL-Divergence Penalty:** An alternative PPO variant uses a penalty on the KL-divergence between the new and old policies. While also effective, the clipped objective is often preferred due to its simpler implementation and robust performance without the need to tune an extra hyperparameter for the KL penalty.

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[r_t(\theta)\hat{A}_t - \beta \text{KL}[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)] \right]$$

  Dynamically adjust the penalty coefficient β, according to $\hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_{\theta_{\text{old}}}(\cdot|s_t)]]$.

4. **Iteration and Convergence**
   The process of data collection (Step 1), advantage estimation (Step 2), and policy optimization (Step 3) is repeated in a loop. With each iteration, the policy π_θ is gradually improved. This training cycle continues until the policy's performance converges or a predefined number of iterations is completed.

Limitations While Proximal Policy Optimization (PPO) has been a foundational algorithm for successfully aligning language models with human preferences, its implementation within the RLHF framework is operationally cumbersome. The multi-stage pipeline, involving reward model fitting and on-policy reinforcement learning loop, is fraught with instability and significant implementation complexity.To address this, Direct Preference Optimization (DPO) offers a paradigm shift.

Direct Preference Optimization[7]

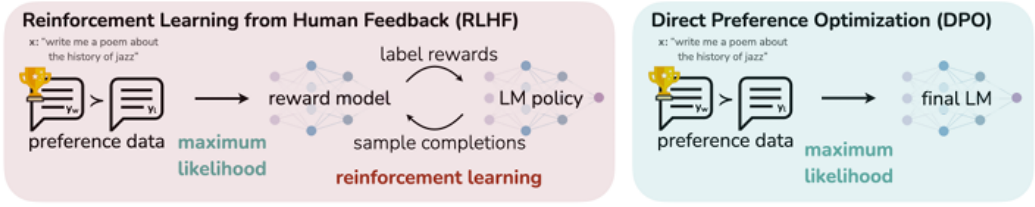


Figure 7. DPO optimizes for human preferences while avoiding reinforcement learning.

Direct Preference Optimization (DPO) presents a paradigm shift from the complex, multi-stage pipeline of traditional RLHF. Its core innovation is the derivation of a direct, analytical mapping between the optimal policy and the underlying reward function, thereby circumventing the need for explicit reward modeling and the instabilities of reinforcement learning.

Theoretical Foundation DPO's breakthrough is the discovery that the reward function $r(x,y)$ constrained by the Bradley-Terry model for human preferences can be expressed in a closed-form relationship with the optimal policy π_r and a reference policy π_{ref}:

$$r(x,y) = \beta \log \frac{\pi_r(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$$

Here, β is a temperature parameter that scales the reward, and $Z(x)$ is a partition function that normalizes the distribution over all possible responses for a given prompt x. This equation reveals that the reward is implicitly defined by the log-probability ratio of a response under the optimal and reference policies.

The DPO Objective Function Leveraging this relationship, DPO formulates a new objective function that allows for direct optimization of the policy on preference data. The policy π_θ is optimized to satisfy the known preference pairs (y_w, y_l) from a dataset $\mathscr{D}$, where y_w is the preferred (winner) response and y_l is the dispreferred (loser) response for a prompt x. The objective function is:

$$\mathscr{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathscr{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) \right]$$

Intuitively, this loss function aims to maximize the likelihood of the preferred responses y_w while minimizing the likelihood of the dispreferred responses y_l. The term inside the sigmoid function is the difference between the estimated rewards for the winning and losing responses, implicitly defined by the policy itself. By maximizing this difference, the policy π_θ directly learns to align with human preferences.

The DPO Training Pipeline The DPO methodology simplifies the traditional RLHF process into a more streamlined, supervised-like training procedure:

1. **Supervised Fine-Tuning (SFT):** A base pre-trained language model is fine-tuned on a high-quality instruction-following dataset. The resulting model serves as the initial reference policy, π_{ref}.
2. **Preference Data Collection:** A dataset $\mathscr{D}$ of human preferences is collected. For each prompt x, multiple responses are generated (often using the SFT model). Human labelers then select the best response (y_w) and the worst response (y_l), forming preference tuples (x, y_w, y_l).
3. **Direct Policy Optimization:** The SFT model π_{ref} is copied to create the policy to be trained, π_θ. This policy is then optimized directly on the preference dataset $\mathscr{D}$ by minimizing the DPO loss function $\mathscr{L}_{\text{DPO}}$. This step is akin to a standard fine-tuning run using an optimizer such as Adam, entirely bypassing the complexities of reinforcement learning. The training continues until convergence or a predefined number of iterations is completed.

Process Supervision

In complex reasoning tasks, a trivial error in a single step might derail the entire logical path, leading to incorrect conclusions. If a model is evaluated solely on the correctness of its final answer (Outcome

Supervision), it becomes nearly impossible to locate the root of the error and correct the flawed reasoning process. This problem is further compounded by the issue of "hallucination" in models trained with RLHF. Such models often learn to produce "convincing wrong answers" that sound plausible but are fundamentally incorrect. Therefore, supervising the reasoning process itself is essential for model reliability.

To meet this need, OpenAI proposed the Process-supervised Reward Models (PRMs)[8]. This approach shifts the focus from the final answer to the individual steps taken to reach it. The models are trained on extensive datasets containing step-level human feedback (800,000 labeled steps for math problems). The method of Active Learning is applied, targeting those "convincing wrong answers". The result is a sophisticated reward model capable of predicting the correctness of each individual step in CoT.

Process Supervision enables exact error localization and quantification of the correctness of each step, leading to significant improvement in model reliability and interpretability. Nevertheless, the high cost of step-wise human annotations and its ability to generalize across diverse domains are still a significant barrier and require future research.

3 Reasoning Large Language Models

OpenAI o1[9]

Building upon the key technologies discussed previously, OpenAI-o1 emerged as a landmark commercial large language model, explicitly engineered for strong reasoning and complex problem-solving. It represented a significant leap from emergent capabilities to intentionally designed reasoning systems.

Inherent CoT Rather than relying on external prompting, o1 internalizes structured CoT as a core component of its response generation. Before delivering an answer, the model engages in a complete, coherent reasoning process. This capability is not an emergent accident but the direct result of **large-scale reinforcement learning**, which refines the model's ability to construct and follow logical steps.

Large-scale Reinforcement Learning Specifically, the training incorporates two critical elements:

- **Process Supervision:** By applying process-based rewards to intermediate steps, the model was explicitly encouraged to learn and favor structured, valid reasoning paths. This directly addresses the need for rewarding the journey, not just the final destination.
- **RLHF & Self-Play:** Through extensive reinforcement learning and self-play, o1 learned to align its internal reasoning process with human preferences. This continuous loop of self-correction and improvement allows it to refine its logic and problem-solving strategies dynamically.

Based on these methodologies, o1 moves beyond simply "predicting the next token" based on learned patterns and human preferences. Instead, it engages in a deliberate, logical, and more interpretable thought process to derive its answers.

Furthermore, o1's performance revealed a **new scaling law**: its reasoning accuracy consistently improves with increased computational "thinking time" during both training and testing. This crucial finding establishes a direct relationship between reasoning time and model capability, offering a new dimension for scaling beyond parameter count.

Despite its advancements, a significant critique against o1 is its hidden reasoning process. While the model performs internal reasoning, this process is not explicitly exposed to the user. This lack of transparency makes it difficult for users to perform their own process supervision or trace the source of an error, countering the open-source spirits of AI community.

DeepSeek-R1[10]

The paper introduces two primary models to enhance LLM reasoning capabilities: DeepSeek-R1-Zero, a pure Reinforcement Learning (RL) approach, and DeepSeek-R1, a more powerful, multi-stage pipeline.

Core Reinforcement Learning Algorithm: GRPO

The foundation for the RL stages in both models is Group Relative Policy Optimization (GRPO), which is chosen for its efficiency. Specifically, for each question q, GRPO samples a group of outputs $o_1, o_2, \cdots, o_G$ from the old policy $\pi_{\theta_{\text{old}}}$ and then optimizes the policy model π_θ by maximizing the following objective:

$$\mathscr{J}_{\text{GRPO}}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}(O|q)]$$
$$\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}A_i, \text{clip}\left(\frac{\pi_\theta(o_i|q)}{\pi_{\theta_{\text{old}}}(o_i|q)}, 1-\varepsilon, 1+\varepsilon\right)A_i\right) - \beta\mathbb{D}_{\text{KL}}(\pi_\theta||\pi_{\text{ref}})\right),$$
$$\mathbb{D}_{\text{KL}}(\pi_\theta||\pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - \log\frac{\pi_{\text{ref}}(o_i|q)}{\pi_\theta(o_i|q)} - 1,$$

where ε and β are hyper-parameters, and A_i is the advantage, computed using a group of rewards $\{r_1, r_2, \ldots, r_G\}$ corresponding to the outputs within each group:

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

Features

- **Critic-less RL:** Unlike standard algorithms like PPO that require a separate, large critic model (Value Model) to estimate advantages, GRPO is "critic-less."
- **Group Sampling:** To calculate the advantage (how much better a specific action is than the average), GRPO samples a group of multiple outputs for a single prompt. It then calculates the mean and standard deviation of the rewards within this group to normalize the rewards and estimate the advantage for each output.
- **Efficiency:** This approach significantly reduces the computational cost and memory overhead of RL training, as it avoids training a large critic network.

Pipeline 1: DeepSeek-R1-Zero

This model was designed to explore if reasoning capabilities can be developed purely through RL without any initial Supervised Fine-Tuning (SFT).

Pipeline (See Figure 8):

1. **Start with a Base Model:** Begin directly with the pre-trained `DeepSeek-V3-Base` model.

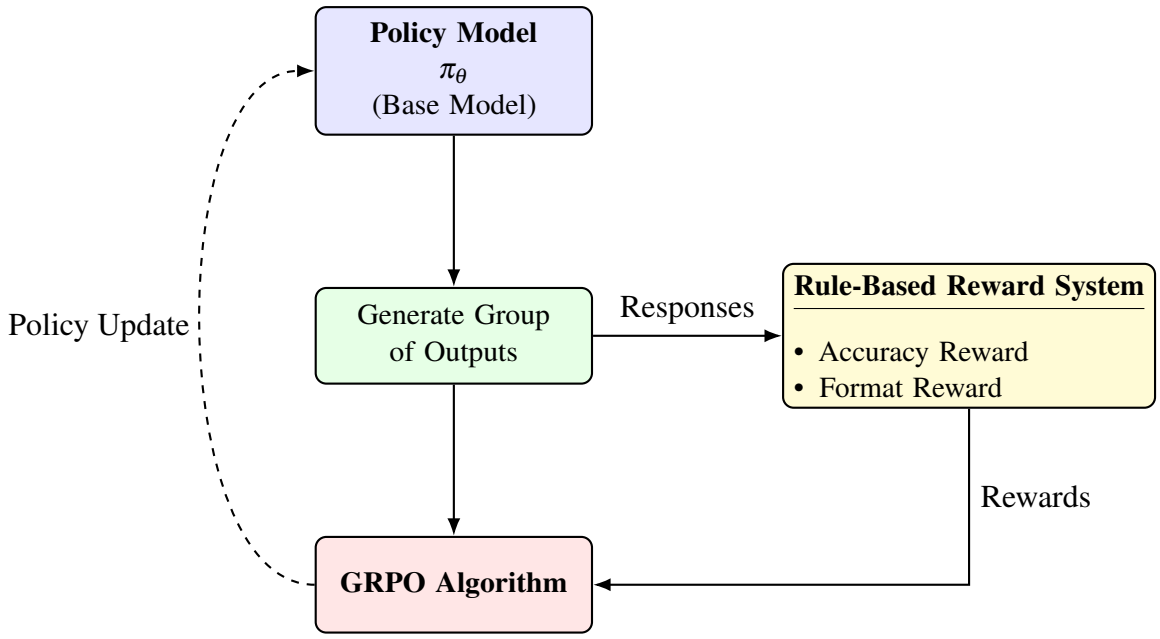


Figure 8. Pipeline of DeepSeek-R1-Zero.

2. **Apply RL with Rule-Based Rewards:** Train the model using the GRPO algorithm. The reward signal is not from a learned model but from a simple, hard-coded Rule-Based Reward System:
    - **Accuracy Reward:** Gives a positive reward if the model's final answer to a problem (e.g. math, coding) is correct. This is verified automatically (e.g. checking a numerical answer or using a compiler).
    - **Format Reward:** Enforces a specific output structure. The model is rewarded for wrapping its reasoning process in `<think>...</think>` tags and its final answer in `<answer>...</answer>` tags.
3. **Outcome:** This pure RL process successfully incentivized complex reasoning behaviors, including long Chain-of-Thought (CoT) and self-reflection (the "aha moment"). However, it suffered from poor readability and language mixing.

Pipeline 2: DeepSeek-R1

This is the main, more powerful model. Its pipeline is designed to fix the shortcomings of R1-Zero and achieve state-of-the-art performance through an iterative process of SFT and RL.

Pipeline (See Figure 9):

1. **Stage 1: Cold Start (Initial SFT)**
   Instead of starting RL from the base model, they first fine-tune `DeepSeek-V3-Base` on a small (a few thousand) but very high-quality dataset of long-form CoT reasoning examples. This effectively prevent the instability of early training and enhance the readability of results.
2. **Stage 2: Reasoning-Oriented RL**
   The model from Stage 1 is trained using the GRPO algorithm, similar to R1-Zero.

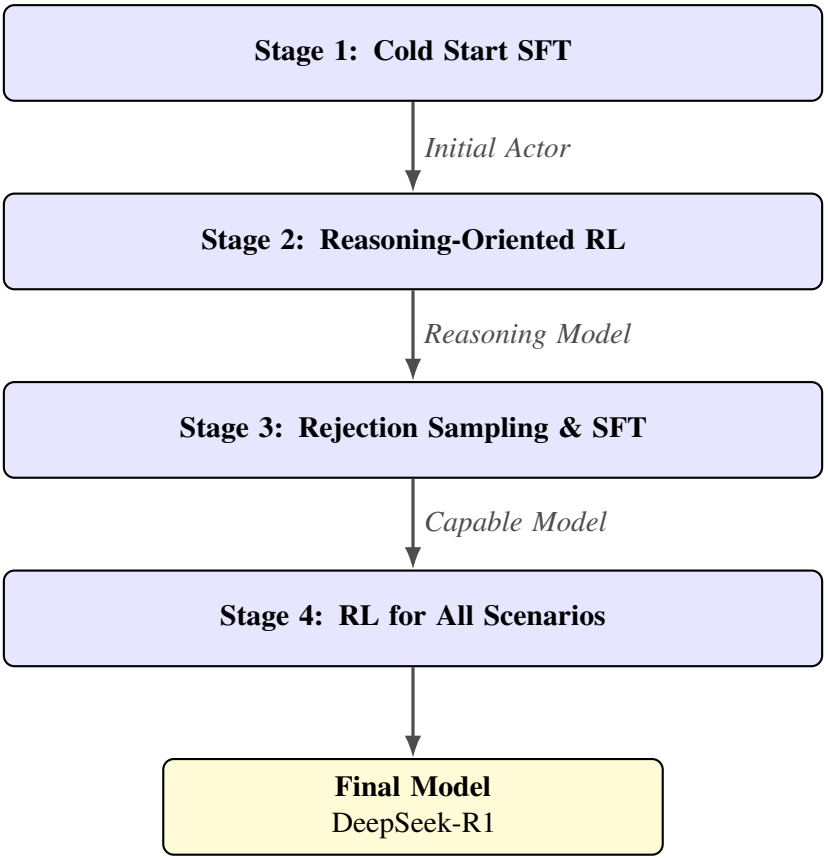


Figure 9. Pipeline for DeepSeek-R1.

Key Addition: A language consistency reward is introduced to solve the language mixing problem. This reward encourages the model to respond in the same language as the prompt.

3. **Stage 3: Rejection Sampling and SFT**
   The improved model from Stage 2 is used to generate a large volume of reasoning data.

   Rejection Sampling: Only the correct and high-quality responses are kept.

   Data Enrichment: Curate both reasoning and non-reasoning data, forming a larger and more diverse dataset.

   New SFT: The base model is fine-tuned again on this curated dataset. This step significantly enhances both reasoning and general capabilities.

4. **Stage 4: RL for All Scenarios**
   The model from Stage 3 undergoes a final RL phase to align it with human preferences across all scenarios.

   Hybrid Rewards: This stage combines rule-based rewards for reasoning tasks and reward models for general tasks to improve helpfulness and harmlessness.

Distillation

A major contribution of the paper is demonstrating the effectiveness of distillation for reasoning.

Goal: To transfer the powerful reasoning capabilities of the large DeepSeek-R1 to smaller, more efficient models.

Process:

1. Use the outputs generated by DeepSeek-R1 to create a large (800k samples), high-quality training dataset.

2. Use this dataset to perform standard Supervised Fine-Tuning (SFT) on smaller open-source models (e.g. Qwen, Llama).

Result: This simple SFT-based distillation is highly effective, enabling smaller models to achieve reasoning performance far beyond what they could by being trained with RL themselves. This shows that the reasoning patterns learned by the large model are highly transferable.

R-Bench[11]

Following the advent of highly capable models such as OpenAI's o1 series and DeepSeek-R1, mainstream reasoning benchmarks are approaching saturation. A critical limitation of these existing evaluation suites is their over-emphasis on knowledge retrieval and factual recognition, while providing insufficient coverage of higher-order cognitive processes like multi-step deduction and the construction of coherent logical chains. Consequently, there is an urgent need for more challenging and nuanced benchmarks to accurately assess the true reasoning capabilities of state-of-the-art models.

To address this evaluation gap, Tsinghua University, in collaboration with other academic institutions, has developed R-Bench. This next-generation benchmark is expressly designed to probe the advanced reasoning abilities of large models. It features a high-quality dataset curated to meet four essential requirements: comprehensiveness, high difficulty, multi-modality, and multi-linguality.

4 Experiment

Objective

The primary objective of this study is to replicate the DeepSeek-R1 training pipeline at a small scale. Through this replication, we aim to provide a clear, empirical analysis of how different training strategies influence the acquisition of reasoning skills.

Experimental Setup

1. **Base Model:** We selected the Qwen2.5-1.5B-Instruct model as our base model. We choose this model for its strong foundational capabilities and manageable size.

2. **Hardware:** All training and evaluation were conducted on the AutoDL platform, utilizing a single NVIDIA RTX 4090 GPU with 24GB of VRAM.

3. **Datasets:** For the "cold-start" phase, we used the HuggingFaceH4/Bespoke-Stratos-17k dataset. For the RL phase, we used the GSM8K dataset.

4. **Reinforcement Learning Configuration:** We utilized the Group Relative Policy Optimization (GRPO) algorithm from the TRL library. For simplicity, we employed a straightforward accuracy-based reward function. A positive reward was granted solely for an exact match between the generated answer and the ground-truth solution.

Evaluation

To assess model performance, we conducted evaluations on two grade-school mathematics problem datasets: GSM8K (in English) and CMATH[12] (in Chinese). For the evaluation, each model was provided with a specific system prompt (See Figure 10) and 8-shot examples for GSM8K (6-shot examples were used for CMATH).

We measured performance against two quantitative metrics:

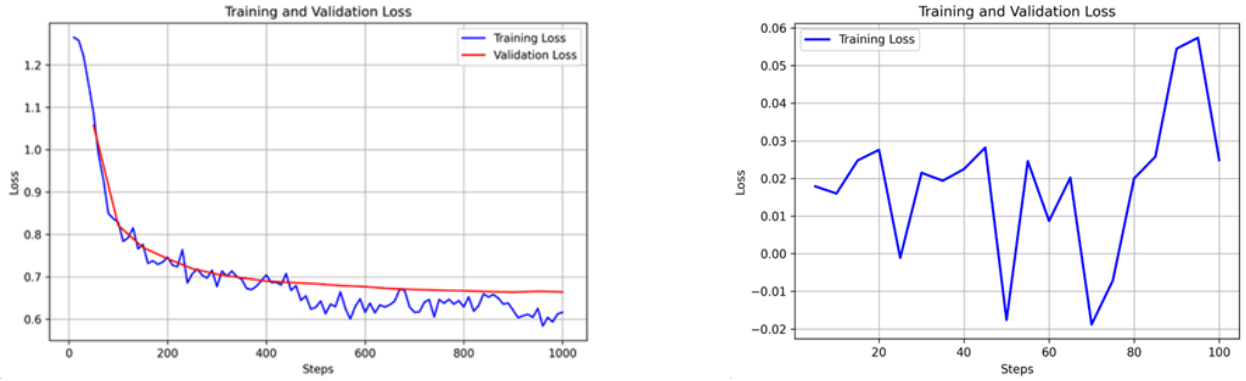
- **Accuracy:** The percentage of problems correctly answered.
- **Consistency:** A measure of whether the model generates a consistent final answer across multiple attempts when using a sampling temperature of 0.5.

```
("You are a helpful assistant that solves math word problems step-by-step. "
 "Please strictly follow the format given in the examples below:\n"
 "Thought: [Your step-by-step reasoning here]\n"
 "Final Answer: [Your final numerical answer here, only the number]\n"
 "Make sure to include both 'Thought:' and 'Final Answer:' in your response, "
 "and ensure the final answer is a SINGLE NUMBER without any additional text or formatting!")
```

Figure 10. System Prompt for Model Evaluation

Results

The loss changes during the training process are presented in Figure 11.



(a) Loss curve of SFT

(b) Loss curve of GRPO

Figure 11. Training Process

Evaluation results are presented in Figure 12. The comparison includes four distinct models:

- **Qwen2.5-1.5B-Instruct:** The base model (instruction-tuned by Qwen).
- **Qwen2.5-1.5B-SFT:** The base model after applying Supervised Fine-Tuning.
- **Qwen2.5-1.5B-SFT-GRPO** The SFT model after additional training with Group Relative Policy Optimization.
- **Qwen2.5-Math-1.5B:** The official math-specialized version of the model.

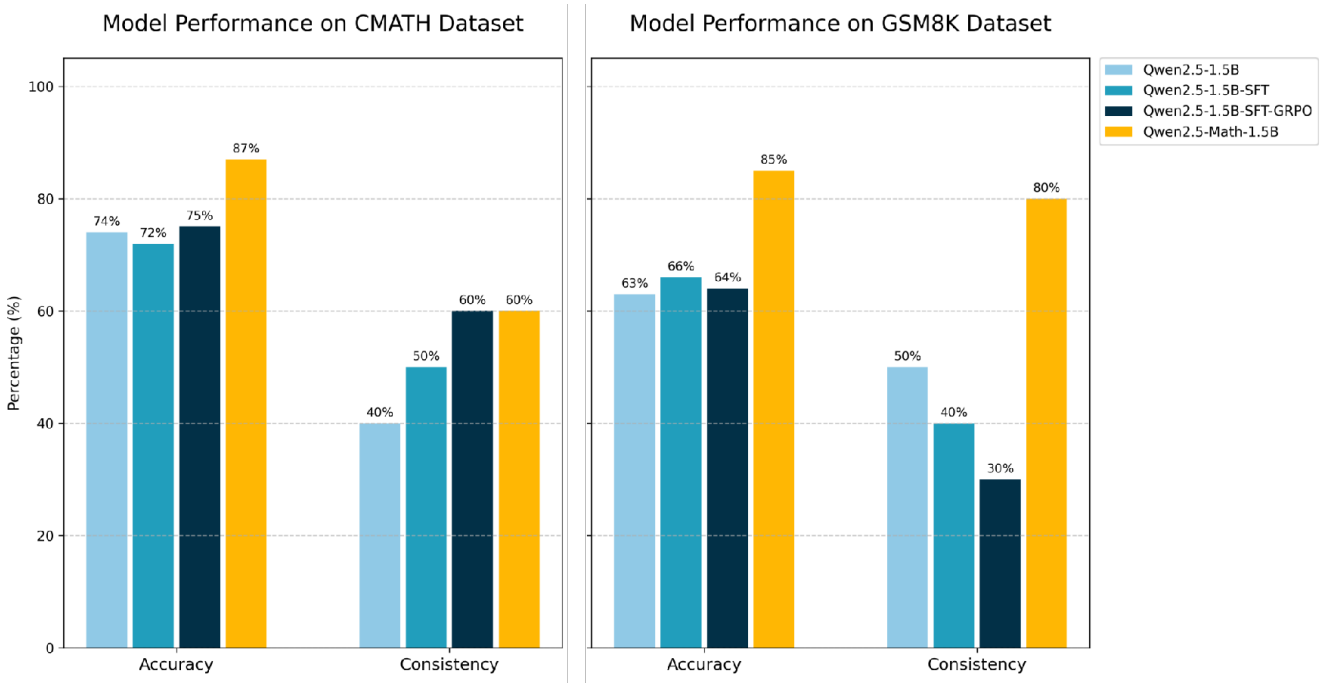


Figure 12. Evaluation Results of Different Models

Discussion

The experimental results (see Figure 12) from both datasets clearly indicate that our SFT and GRPO training pipeline failed to yield significant improvements in accuracy. To diagnose the root cause of this underperformance, we examined the training dynamics of each phase.

As shown in Figure 11, the Supervised Fine-Tuning (SFT) phase exhibited a typical, steadily decreasing loss, indicating successful convergence. In stark contrast, the loss curve for the Group Relative Policy Optimization (GRPO) phase demonstrated significant instability and failed to converge. To address this non-convergence, we conducted several experiments aimed at stabilizing the training process. These included systematically reducing the learning rate from 5e-5 to 1e-5, and subsequently to 5e-6, as well as simplifying the reward function to the aforementioned accuracy-based signal. However, despite these interventions, the convergence issue persisted, suggesting the instability may stem from more fundamental aspects of our experimental setup.

Upon reflection of our initial experiments, we hypothesized that the ineffectiveness of our training stemmed from the choice of the base model. Multiple research suggests that the performance ceiling of a language model is largely determined by its pre-training, which indicating the improvement of RL is limited. Given

that our base model, Qwen2.5-1.5B-Instruct, has already undergone extensive instruction fine-tuning, its performance was likely near a saturation point, leaving minimal room for further improvement through our methods.

Based on this analysis, we revised our approach by switching the base model to Qwen2-1.5B, a non-instruction-tuned model. Disappointingly, this led to a significant degradation in model performance immediately following the Supervised Fine-Tuning (SFT) phase—an issue not observed with the original Instruct model. We hypothesize that the base Qwen2-1.5B model, lacking the robustness conferred by instruction tuning, is more susceptible to the "catastrophic forgetting" of pre-trained knowledge during full-parameter SFT.

Consequently, we attempted two countermeasures to mitigate this issue: (1) substantially reducing the volume of the SFT training data, and (2) employing Low-Rank Adaptation (LoRA) to finetune. However, neither of these interventions succeeded in preserving the model's foundational capabilities.

In summary, this series of experiments highlights the non-trivial challenges of improving the reasoning ability of small-scale language models. Our findings reveal a critical trade-off: while a pre-instruction-tuned model like Qwen2.5-1.5B-Instruct is robust, it appears too saturated to benefit from reinforcement learning. Conversely, a non-tuned base model like Qwen2-1.5B, while theoretically having more room for improvement, proved too "fragile" for SFT, suffering from catastrophic forgetting. This suggests that a base model with a larger parameter count may be a prerequisite for successfully navigating this fine-tuning stage.

Ultimately, our work underscores that the success of improving the reasoning ability is not merely about applying advanced techniques and following successful training pipelines, but is deeply dependent on the interplay between the base model's initial state and the fine-tuning methodology. Future research should therefore focus on two key areas: (1) developing more stable fine-tuning methods that can adapt a base model without compromising its foundational knowledge, and (2) investigating the parameter threshold at which models become robust enough for effective reasoning alignment.

5 Limitations & Future Directions

While AI community has been witnessing remarkable progress with incredible speed, the path to truly robust and reliable AI is still marked by significant challenges. Addressing these limitations is the focus of current and future research, aiming to build models that are not only more capable but also more trustworthy, efficient, and grounded in reality.

The Paradox of Hallucination

Hallucination is a persistent challenge in advanced reasoning models. Reports from models like o3 and o4-mini suggests improvements in reasoning capabilities even exacerbate hallucination rather than prevent it. This severely limits the application of reasoning models in critical domains such as healthcare or financial decision-making, where reliability is non-negotiable.

Promising future directions include:

- **External Fact-Checking & Claiming:** Integrating mechanisms that verify intermediate reasoning steps against trusted external knowledge sources.

- **Uncertainty Quantification:** Developing models that can express their own confidence about a conclusion, signaling when human oversight is required.
- **Strict Human/AI Feedback:** Implementing more rigorous feedback loops during training that heavily penalize any deviation from factual accuracy.

Computational Cost and Latency

Advanced reasoning techniques comes at a significant computational cost, leading to high latency and resource demands. Making these powerful models accessible and practical for real-time applications is a critical engineering hurdle. Current strategies to address this include:

- **Hybrid Reasoning:**Methods like "thinking budgets" adopted by Gemini 2.5 Flash enables the model to dynamically allocate computational resources based on problem difficulty.
- **Knowledge Distillation:** Training smaller, more efficient models to mimic the reasoning capabilities of larger ones (e.g. Deepseek-R1-Distill-Qwen-7B).
- **Efficient Architecture and Hardware Acceleration:** The continuous development of novel model architectures and specialized hardware remains a fundamental path toward reducing the cost and time of complex reasoning.

Multimodal Reasoning & Real-World Understanding

Real-world Understanding is one of the ultimate goal of AI researchers. Current reasoning LLMs are mostly limited to abstract symbolic reasoning and are still far from understanding real, physical world. To evolve towards a true "world model," AI must bridge this gap. This requires progress in several key areas:

- **Richer Multimodal Datasets:** Moving beyond text-only training to incorporate diverse data modalities like images, sounds, and other sensory experiences to build a more holistic understanding.
- **Embodied AI and Interactive Learning:** Developing agents that can learn from direct interaction with an environment. This is essential for grasping fundamental concepts like **cause and effect**, **physical properties of objects**, and intuitive **spatial reasoning**.

Interpretability and Safety

The long-standing yet critical concerns of interpretability and safety (including ethical considerations) become even more urgent as models become more autonomous. As models perform complex, multi-step reasoning, understanding how they arrive at a conclusion is vital for debugging, trust, and accountability. Likewise, ensuring their behavior remains aligned with human values is paramount to their responsible deployment.

Conclusion

In summary, the evolution from chatbots, reasoners to Artificial General Intelligence (AGI) is an ongoing journey. The future of generative AI reasoning lies not only in enhancing model intelligence, but also in making that intelligence reliable, efficient, and fundamentally safe for real-world use.

References

1. Jason Wei, D. S. M. B. F. X. E. C. Q. L. D. Z., Xuezhi Wang. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903* DOI: 10.48550/arXiv.2201.11903 (2022).
2. Takeshi Kojima, M. R. Y. M. Y. I., Shixiang Gu. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, vol. 35, 22199–22213 (2022).
3. Shunyu Yao, J. Z. I. S. T. L. G. Y. C. K. N., Dian Yu. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601* DOI: 10.48550/arXiv.2305.10601 (2023).
4. Yongchao Zhou, Z. H. K. P. S. P. H. C. J. B., Andrei Ioan Muresanu. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910* (2022). 2211.01910.
5. Long Ouyang, X. J. D. A. C. L. W. P. M. C. Z. S. A. K. S. A. R. J. S. J. H. F. K. L. M. M. S. A. A. P. W. P. C. J. L. R. L., Jeff Wu. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155* DOI: 10.48550/arXiv.2203.02155 (2022).
6. John Schulman, P. D. A. R. O. K., Filip Wolski. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017). 1707.06347.
7. Rafael Rafailov, E. M. S. E. C. D. M. C. F., Archit Sharma. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18954* (2024).
8. Hunter Lightman, Y. B. H. E. B. B. T. L. J. L. J. S. I. S. K. C., Vineet Kosaraju. Let's verify step by step. *arXiv preprint arXiv:2305.20050* (2023).
9. OpenAI. Learning to reason with llms (2024).
10. Guo, D. *et al.* Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025). 2501.12948.
11. Meng-Hao Guo, e. a. R-bench: Graduate-level multi-disciplinary benchmarks for llm&mllm complex reasoning evaluation. *arXiv preprint arXiv:2501.04412* (2025).
12. Tianwen Wei, e. a. Can your language model pass chinese elementary school math test? *arXiv preprint arXiv:2309.04953* (2023).

Author contributions statement

- **Junwei Huang:**
  - CoT, Self-Improving Prompts, Process Supervision, OpenAI o1, Limitations and Future work.
  - Conducted experimental evaluation and result analysis.
- **Chenyue Fang:**
  - RLHF, PPO, DPO, DeepSeek-R1.
  - Implemented the experimental pipeline and wrote the discussion section analyzing the experimental findings.