BIG JUDGES, SMALL SOLVERS: REWARDING CHAIN-OF-THOUGHT QUALITY IN SMALL LANGUAGE MODELS

Abhilash Abhilash (SBU ID: 115907810)
abhilash.abhilash@stonybrook.edu

Ananya Sadana (SBU ID: 115799974)
ananya.sadana@stonybrook.edu

Sumedh Ghavat (SBU ID: 115854819)
sumedh.ghavat@stonybrook.edu

ABSTRACT

Small language models often guess the right number while producing shaky logic, and binary correctness rewards cannot teach them to reason. We tackle this by *verifying the chain-of-thought*: a 70-B parameter judge (DeepSeek-R1) scores each rationale on a three-level scale and feeds that signal into critic-free GRPO fine-tuning of a 4-B parameter Gemma student. Two schedules are explored — **Method 1** queries the judge only when the answer is correct, whereas **Method 2** also grants partial credit when the answer is wrong but the steps make sense. The approach lifts @1 accuracy to 81.3 % on GSM-8K (+2.4 pp over a binary-reward baseline) and to 33.1 % on the harder MATH-500 (+5.8 pp), with Method 2 reaching the same score in half as many updates. A simple style-length penalty in the prompt curbs reward hacking without extra computation. These results show that reasoning-aware verifier rewards make small models both more transparent and more sample-efficient, opening a path toward low-compute tutoring and domain-specific reasoning agents.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated impressive capabilities in complex natural language processing tasks, including arithmetic and symbolic reasoning. Despite these advances, smaller LLMs often produce correct answers through flawed or incoherent reasoning processes. This reliance on shortcut heuristics rather than genuine multi-step reasoning poses a significant barrier to their deployment in domains that demand interpretability and reliability, such as education and scientific computation.

Recent reinforcement learning techniques, particularly Reinforcement Learning from Human Feedback (RLHF), have shown promise in aligning LLM outputs with desired behavior. However, these methods typically rely on scalar or binary correctness signals, which offer no insight into the reasoning that underlies a given answer. For tasks like mathematics, where the process leading to the answer is as important as the answer itself, such coarse feedback fails to encourage the development of robust intermediate reasoning.

To address this limitation, we propose a framework that augments reward signals using a large verifier model capable of assessing the quality of the chain-of-thought. Our approach employs a 70B-parameter LLM (DeepSeek-R1) as an external judge that evaluates the student models solution on a three-point rubric, scoring both correctness and the reasoning trajectory. The resulting multi-level reward is then used to fine-tune a much smaller student model (Gemma-3-4B) through Group Relative Policy Optimization (GRPO), a stable and critic-free reinforcement learning algorithm.

We evaluate our methods on two prominent math benchmarks: GSM-8K, which consists of arithmetic word problems, and MATH-500, a symbolic reasoning dataset drawn from mathematics competitions. Our findings suggest that incorporating reasoning-aware rewards not only improves accuracy but also accelerates learning, especially for tasks that require multi-step logic. These results highlight the importance of rewarding process over product and point toward future avenues for low-compute, high-transparency reasoning agents.

In summary, our work makes the following contributions:

1. We introduce a verifier-based, multi-level reward function that evaluates the quality of chain-of-thought reasoning, allowing small models to receive partial credit for logically valid but incomplete solutions.
2. We integrate this reward into a stable and efficient fine-tuning pipeline based on Group Relative Policy Optimization (GRPO), combined with parameter-efficient methods such as QLoRA and Unsloth to enable training on a single GPU.
3. We empirically validate our approach on GSM-8K and MATH-500, showing improvements in accuracy and sample efficiency over binary-reward baselines, and conduct ablation analyses to understand the effects of different reward schedules and verifier models.

2 Related Work and Background

Reinforcement learning has emerged as a powerful tool for aligning large language models (LLMs) with human preferences. Reinforcement Learning from Human Feedback (RLHF), popularized by InstructGPT (Ouyang et al., 2022), uses reward models trained on human-labeled preferences to fine-tune LLMs. While RLHF improves helpfulness and safety, its reliance on scalar or binary rewards limits its ability to guide complex reasoning. A reward of 0 or 1 reveals little about the validity of intermediate steps, especially in tasks requiring multi-step logic.

To overcome this limitation, recent research has focused on Reinforcement Learning with Verifiable Rewards (RLVR), which leverages task-specific verifiers to assess both correctness and coherence. In structured domains like mathematics and programming, where outputs can be programmatically validated, such verifiers provide more nuanced supervision. DeepSeek-R1 (Guo et al., 2025) exemplifies this by using exact-match evaluation for math answers. However, it stops short of assessing the reasoning chain itself, leaving the model's internal logic unchecked.

This has prompted interest in chain-of-thought (CoT) prompting (Wei et al., 2022), which encourages models to explicitly decompose complex problems. CoT has shown promise in improving accuracy in zero-shot settings, yet it lacks a mechanism to assess the quality of the generated reasoning, making it prone to hallucinated or invalid logic.

Policy optimization techniques also play a critical role in enabling stable fine-tuning. Group Relative Policy Optimization (GRPO), introduced in the DeepSeek framework (Guo et al., 2025), improves over Proximal Policy Optimization (PPO) (Schulman et al., 2017) by eliminating the need for a learned critic. Instead, GRPO compares multiple sampled trajectories using normalized groupwise rewards, offering enhanced training stability and sample efficiency–particularly useful when value estimation is noisy.

Simultaneously, parameter-efficient fine-tuning (PEFT) methods have reduced the resource demands of LLM training. Low-Rank Adaptation (LoRA) (Hu et al., 2021) adds trainable low-rank matrices to frozen model weights, enabling efficient adaptation with fewer parameters. Quantized LoRA (QLoRA) (Dettmers et al., 2023) extends this by applying 4-bit quantization, allowing even billion-parameter models to be fine-tuned on consumer-grade GPUs. These techniques have significantly democratized access to reinforcement learning and instruction tuning for smaller labs and individuals.

Our work builds on these trends by integrating verifier-based reasoning rewards, chain-of-thought prompting, GRPO optimization, and parameter-efficient fine-tuning to train small models that not only produce correct answers but also exhibit interpretable reasoning.

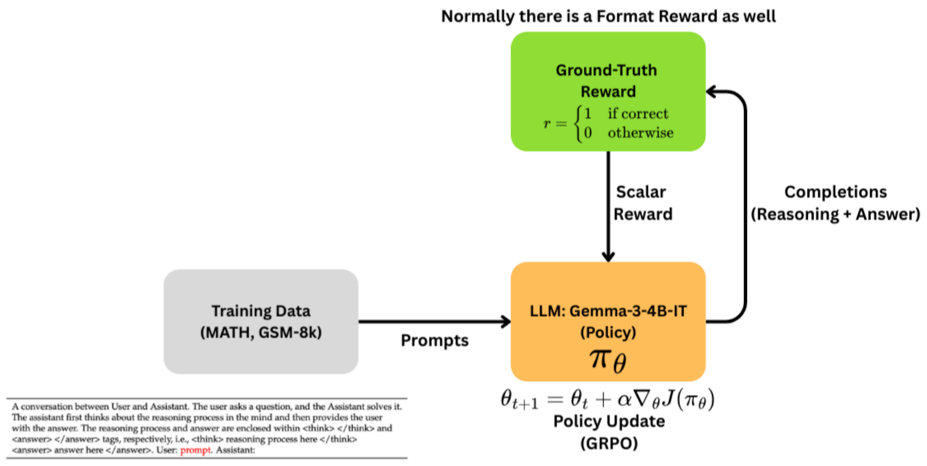
3 Method

3.1 Problem Setup

We consider a standard reinforcement learning framework in which a small language model, referred to as the student, learns to solve math problems by generating a trajectory $\tau = (\text{CoT}, \text{Answer})$ that includes both a chain-of-thought and a final numeric or symbolic result. The policy π_θ defines a distribution over such trajectories conditioned on the input question. The learning objective is to maximize the expected reward over trajectories sampled from this policy.

In the binary-reward baseline, a trajectory receives a reward $r_{\text{ans}} = 1$ if the final answer is correct and 0 otherwise. While effective for simple arithmetic tasks, this reward provides no guidance when the answer is incorrect or when reasoning quality varies across samples with correct answers.

Figure 1 illustrates this basic setup, where the student model receives a scalar reward based solely on the correctness of the final answer.


Normally there is a Format Reward as well
Ground-Truth Reward
$r = \begin{cases} 1 & \text{if correct} \\ 0 & \text{otherwise} \end{cases}$
Scalar Reward
Completions (Reasoning + Answer)
Training Data (MATH, GSM-8k)
Prompts
LLM: Gemma-3-4B-IT (Policy)
π_θ
$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\pi_\theta)$
Policy Update (GRPO)

A conversation between User and Assistant. The user asks a question, and the Assistant solves it. The assistant first thinks about the reasoning process in the mind and then provides the user with the answer. The reasoning process and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively, i.e., <think> reasoning process here </think> <answer> answer here </answer>. User: prompt. Assistant:

Figure 1: Baseline GRPO training with binary ground-truth reward for the final answer.

3.2 Verifier-Based Reward Function

To provide more informative feedback, we introduce a three-level scoring function for reasoning quality, denoted $r_{\text{cot}} \in \{0, 1, 2\}$. This score is generated by a large teacher model (DeepSeek-R1-70B) that evaluates the coherence, relevance, and correctness of the students chain-of-thought. The final reward for a trajectory is computed as a weighted sum:

$$r = \alpha \cdot r_{\text{ans}} + \beta \cdot r_{\text{cot}},$$

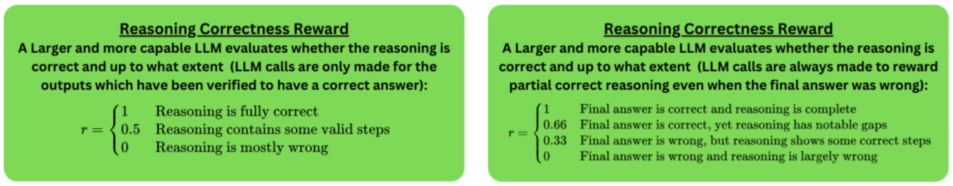
where α and β balance the emphasis on answer correctness versus reasoning quality. This reward is used to update the student policy through GRPO.

We investigate two reward schedules:

- **Method 1 (Selective Evaluation):** The verifier is queried only when the students final answer is correct. This setting is cost-efficient and discourages unnecessary feedback for obviously incorrect completions.
- **Method 2 (Full Evaluation):** The verifier scores every completion, even those with wrong answers. This approach grants partial credit for logically valid intermediate steps, fostering more robust learning at the expense of additional compute.

Figure 2 provides a side-by-side view of the reward rubrics used in both methods.

To ensure that the verifier's judgments reflect genuine reasoning quality, we design prompts that penalize incoherent or overly verbose explanations. This guards against reward hacking by discouraging stylistic padding. The full prompt texts for both reward schedules can be found in Appendix A (Method 1) and Appendix B (Method 2).



Reasoning Correctness Reward

A Larger and more capable LLM evaluates whether the reasoning is correct and up to what extent (LLM calls are only made for the outputs which have been verified to have a correct answer):

$$r = \begin{cases} 1 & \text{Reasoning is fully correct} \\ 0.5 & \text{Reasoning contains some valid steps} \\ 0 & \text{Reasoning is mostly wrong} \end{cases}$$

Reasoning Correctness Reward

A Larger and more capable LLM evaluates whether the reasoning is correct and up to what extent (LLM calls are always made to reward partial correct reasoning even when the final answer was wrong):

$$r = \begin{cases} 1 & \text{Final answer is correct and reasoning is complete} \\ 0.66 & \text{Final answer is correct, yet reasoning has notable gaps} \\ 0.33 & \text{Final answer is wrong, but reasoning shows some correct steps} \\ 0 & \text{Final answer is wrong and reasoning is largely wrong} \end{cases}$$

Figure 2: Reasoning reward rubric for Method 1 (left) and Method 2 (right).

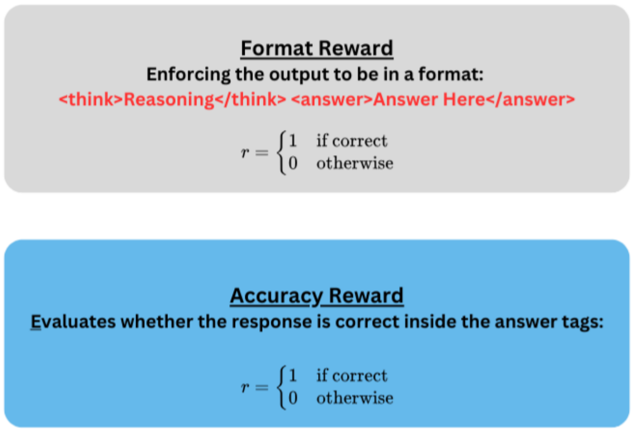
3.3 Policy Optimization via GRPO

The training process follows the Group Relative Policy Optimization (GRPO) algorithm. For each input question, K candidate trajectories are sampled from the current policy. Each trajectory is scored using the reward function described above. The rewards are then normalized within the group to compute relative preferences. The GRPO objective for a batch of B questions is:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{B}\sum_{j=1}^{K} \frac{\exp(r_{ij})}{\sum_{k=1}^{K}\exp(r_{ik})} \log \pi_\theta(\tau_{ij})$$

where r_{ij} is the reward for the j-th trajectory of the i-th input. This objective encourages the model to increase the probability of higher-ranked trajectories without relying on a value function or critic.

To support reasoning-aware optimization, we also incorporate format and accuracy constraints into the reward. These are illustrated in Figure 3, which shows the components of the full reward function: format correctness and answer correctness.



Format Reward

Enforcing the output to be in a format:

<think>Reasoning</think> <answer>Answer Here</answer>

$$r = \begin{cases} 1 & \text{if correct} \\ 0 & \text{otherwise} \end{cases}$$

Accuracy Reward

Evaluates whether the response is correct inside the answer tags:

$$r = \begin{cases} 1 & \text{if correct} \\ 0 & \text{otherwise} \end{cases}$$

Figure 3: Components of the baseline reward: format correctness (top) and accuracy (bottom).

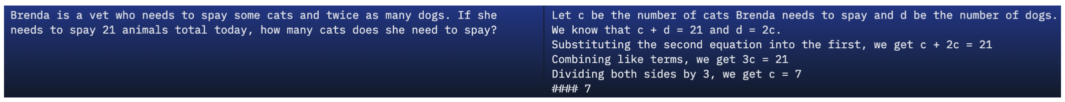
By combining verifier-based rewards with a stable and sample-efficient optimization strategy, this method enables small models to acquire interpretable reasoning skills under limited computational budgets.

4 Datasets & Task Setup

GSM8K. The Grade-School Math 8K corpus (Cobbe et al., 2021) contains 7473 train and 1319 test word-problems that require onetwo arithmetic operations. Each example is a plain-text question followed by an author-provided chain-of-thought and a final line `$$<ans>$$` holding the numeric answer. During fine-tuning we show the student *only the question*, expect it to emit reasoning plus answer, and compute reward with the official exact-match checker. Figure 4a illustrates a typical item.

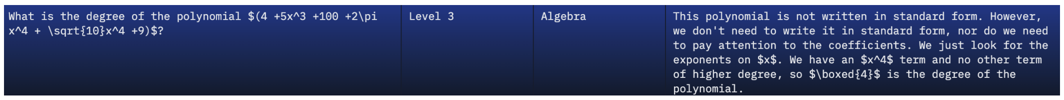
MATH-500. We evaluate on the 500-problem competition subset of the MATH benchmark (Hendrycks et al., 2021). Questions span algebra, geometry, combinatorics, and number theory and often need 510 symbolic steps. Ground-truth answers are symbolic expressions; we mark correctness with `sympy` equivalence. An example appears in Figure 4b.

Prompt format and context length. Each question is framed as a chat turn; the student must reply with `<think></think>` for its chain-of-thought and `<answer></answer>` for the final result. Any other layout receives zero reward. Prompt (system + user) is capped at 384 tokens and the combined `think` + `answer` region at 640, so the full sequence never exceeds 1024 tokens, consistent with Section 5. The contents of `think` are hidden at test time; only the `answer` tag is evaluated.



Brenda is a vet who needs to spay some cats and twice as many dogs. If she needs to spay 21 animals total today, how many cats does she need to spay?	Let c be the number of cats Brenda needs to spay and d be the number of dogs. We know that c + d = 21 and d = 2c. Substituting the second equation into the first, we get c + 2c = 21 Combining like terms, we get 3c = 21 Dividing both sides by 3, we get c = 7 #### 7

(a) GSM8K sample



What is the degree of the polynomial $(4 +5x^3 +100 +2\pi x^4 + \sqrt{10}x^4 +9)$?	Level 3	Algebra	This polynomial is not written in standard form. However, we don't need to write it in standard form, nor do we need to pay attention to the coefficients. We just look for the exponents on x. We have an x^4 term and no other term of higher degree, so $\boxed{4}$ is the degree of the polynomial.

(b) MATH-500 sample

Figure 4: Example problems from the two evaluation datasets.

5 Experimental Details

5.1 Models

- **Student (policy).** We fine-tune `gemma-3-4b-it`, an open-source model that fits on a single Colab GPU after 4-bit quantisation.
- **Teacher (verifier).** We query `DeepSeek-R1-70B` through Together AIs API to score the students chain-of-thought on a 0/1/2 scale. DeepSeek was chosen for its strong math reasoning; an alternative verifier is evaluated in Section 7.2.

5.2 Training Setup

- **Parameter-efficient fine-tuning.** We use the Unsloth library with QLoRA: 4-bit weight quantisation and LoRA rank 32 adapters. Only language layers, attention blocks and MLPs are trainable.
- **Hardware.** All runs fit in 23 GB GPU memory (NVIDIA A100-80 GB or L4-24 GB on Colab Pro). A 1000-step run takes about 24 h.
- **Algorithm.** Training follows the critic-free GRPO procedure. For each minibatch we generate 6 candidate chains, call the verifier once per candidate, and apply the GRPO update.

5.3 Key Hyper-parameters

Discussion of Table 1. The listed hyper-parameters are the only ones we found to have a noticeable impact on performance or compute budget. Most values follow common LoRA or GRPO defaults; the main exception is the KL-regularisation weight β. We began with β=0.004 (the value in the original DeepSeek release) but found that the stronger penalty froze exploration and slowed reward improvement. Lowering it to β=0.001 loosened the trust-region constraint, producing smoother training curves and higher validation accuracy, so we use that setting for all results reported in Section 6.

Parameter	Symbol / Flag	Value
Context length	L_{max}	1 024 tokens
Learning rate	η	5×10^{-6}
Optimizer	—	`adamw_torch_fused`
LoRA rank	r	32
Quantisation	bits	4-bit (NF4)
Per-device batch size	B	1 sample
Gradient accumulation steps	n_{acc}	6
Generations per prompt	k	6
GRPO KL weight	β	0.001
Epochs	N_{ep}	5

Table 1: Core hyper-parameters for all main experiments.

6 Results

6.1 Test-set Accuracy

We evaluate the baseline RLVR system (binary reward) and our **Method 1** (verifier reward 0/1/2) on the official test splits of MATH-500 and GSM-8K. All numbers are *@1* accuracies, meaning a sample is counted correct only if the *first* generated answer string matches the ground truth.[1]

	MATH-500@1	**GSM-8K@1**
Gemma-3-4B-IT	10.71%	25.57%
Baseline Model (RLVR)	27.32%	78.92%
Our Model Method 1 (RLVR)	**33.11%**	**81.31%**
Our Model Method 2 (RLVR)	33.05%	

Table 2: Test-set accuracy (%); higher is better.

Key observations.

- **Dataset difficulty matters.** GSM-8K problems usually require only one- or two-step arithmetic, so Method 1 improves accuracy by a modest $+2.4$ pp ($78.9 \rightarrow 81.3$ %). In contrast, MATH-500 demands multi-step symbolic reasoning; under the same verifier reward, accuracy climbs by $+5.8$ pp ($27.3 \rightarrow 33.1$ %). This supports our hypothesis that chain-of-thought scoring is *most* helpful when correct answers cannot be reached via simple shortcut heuristics.
- **Method 2 gives richerbut costlierfeedback.** Method 2 queries the verifier on *every* generated answer, so it also awards partial credit for good reasoning but wrong final answer. Even though we could afford only 500 GRPO steps (half of Method 1) due to the extra API calls, it still reaches 33.0 % on MATH-500, matching Method 1s full run. The denser reward signal therefore accelerates learning, albeit at higher compute cost.

Ablation plots illustrating sample efficiency appear in Section 7.

7 Analysis & Ablations

7.1 Training Dynamics and Sample Efficiency

Figure 5 plots the mean reward received by the student policy during training.

- **GSM-8K.** Both curves rise at almost the same pace and saturate after ∼400 updates. This echoes the small accuracy gap seen in Table 2: when problems demand only onetwo arithmetic steps,

[1]For GSM-8K we use exact string match; for MATH-500 we follow Hendrycks et al. (2021) and evaluate equality after `sympy` simplification.

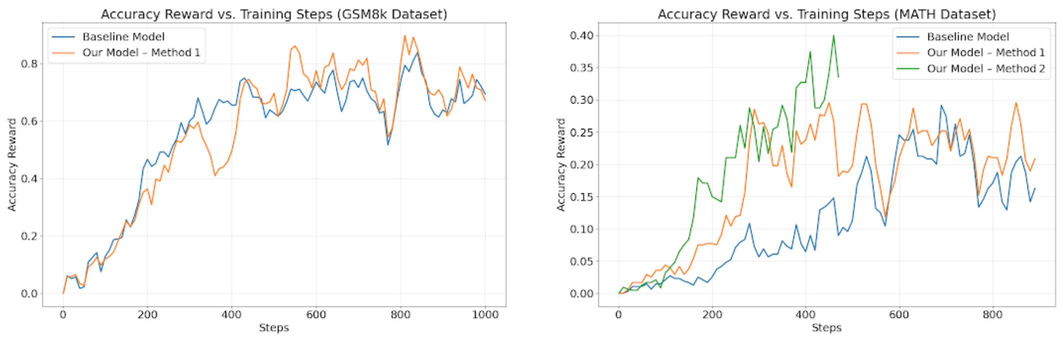


Figure 5: Average accuracy reward vs. GRPO training steps. **Left:** GSM-8K. **Right:** MATH-500.

binary correctness already provides a strong learning signal, so additional reasoning feedback offers little advantage.

- **MATH-500.** Method 1 climbs faster than the baseline, and Method 2 is even steeper, reaching a comparable reward level in roughly half the steps. The denser, reasoning-aware reward thus improves *sample efficiency*the key metric in RL fine-tuning, where final convergence may be similar, but training time and stability are paramount.

7.2 Teacher-Model Ablation

To test how much the verifiers quality matters, we replaced `DeepSeek-R1-70B` with `Llama-3.3-70B-instruct-turbo` and reran Method 1 for 200 GRPO steps. The reward curve rose markedly slower and plateaued below the DeepSeek run. Given the poor trajectory we did not invest further GPU time.

These observations suggest that (i) the mathematical reasoning strength of the teacher is crucial and (ii) inferior verifiers degrade the reward signal, erasing the gains reported in Section 6. Therefore, selecting or training a high-quality judge remains an important practical constraint for verifier-based RL.

8 Conclusion

We introduced two reasoning-aware reward functions for critic-free GRPO fine-tuning of a 4-billion-parameter Gemma model. **Method 1** calls the teacher LLM only when the students numeric answer is already correct, assigning $r \in \{1, 0.5, 0\}$ according to the quality of the accompanying chain-of-thought; **Method 2** evaluates *every* sample assigning $r \in \{1, 0.66, 0.33, 0\}$ so the student can still earn partial credit when the final answer is wrong but intermediate steps are sound. Both schedules raise test accuracy on the multi-step MATH-500 set, with Method 1 delivering +5.8 pp and Method 2 matching that gain in only half the updates, while the simpler GSM-8K sees a smaller yet consistent +2.4 pp boost. Faster learning curves confirm that denser, reasoning-based feedback improves sample efficiency, especially when shortcut heuristics no longer suffice. A style-length penalty in the judge prompt mitigates reward hacking at negligible cost. These findings highlight the practical value of verifying chains of thought and point toward future work on cheaper, domain-specific verifiers and extensions to code or scientific-reasoning tasks.

References

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL `https://arxiv.org/abs/2110.14168`.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and et al. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023. URL `https://arxiv.org/abs/2305.14314`.

Daya Guo, Dejian Yang, Haowei Zhang, and Junxiao Song. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL `https://arxiv.org/abs/2501.12948`.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL `https://arxiv.org/abs/2103.03874`.

Edward Hu, Yelong Shen, Phillip Wallis, and et al. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. URL `https://arxiv.org/abs/2106.09685`.

Long Ouyang, Jeff Wu, Xu Jiang, and et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022. URL `https://arxiv.org/abs/2203.02155`.

John Schulman, Filip Wolski, Prafulla Dhariwal, and et al. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL `https://arxiv.org/abs/1707.06347`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, and et al. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022. URL `https://arxiv.org/abs/2201.11903`.

A FULL JUDGE PROMPT FOR METHOD 1

```
You are an impartial grader of mathematical reasoning.
For each submission you will receive exactly two items:

1. Question
2. Student reasoning that claims to reach the correct final answer
   (The numeric answer itself has already been verified.)

Your job: read the student's reasoning, decide how sound it is,
and assign a single integer score using the rubric below.

Rubric
0 - Reasoning is mostly wrong, off-topic, or unverifiable.
1 - Reasoning contains some valid steps but also gaps, unjustified
leaps, or errors.
2 - Reasoning is fully correct, logically complete, and properly
justified.

Style penalty
If the writing is flashy/ornate yet mathematically incoherent, subtract
**1 point** after applying the rubric (minimum score = 0).

Note: Do not penalize the students reasoning if it first went in the
wrong direction initially and then corrected itself, that is actually a
good sign.

Output format (strict)
Reply with nothing except <score>n</score> where n can be only from the
set {0, 1, 2}.
Do not output anything else.
```

B FULL JUDGE PROMPT FOR METHOD 2

```
You are an impartial grader of mathematical reasoning who needs to
evaluate the students answer and reasoning for the provided
question. You will also be provided the ground truth correct
answer for the question, so you dont need to worry about that.

For every submission you will receive three items (always in this
order):

1. Question
2. Correct Answer (ground-truth numeric answer)
3. Student Reasoning that claims to reach the final answer

Your tasks, in order
--------------------
1. Check the student s final answer against the provided Correct
(ground truth) Answer. If you cant find the students answer then
consider it as wrong for the scoring criteria atleast.
2. Evaluate the reasoning for logical soundness, completeness, and
mathematical validity.
3. Assign a single integer score using the rubric below, then
apply the style penalty if needed.

Important Note: Do not start solving the question on your own, you
just need to evaluate the reasoning and the final answer of the
student and not your own.

Scoring Rubric (before any style penalty)
---------------------------------------------------------------
| Score | When to give it                                      |
|-------|------------------------------------------------------|
| 0 | Final answer is wrong and reasoning is largely wrong, off-
topic, incoherent, or unverifiable.            |
| 1 | Final answer is wrong, but reasoning shows some correct
steps, ideas, or partial progress.                 |
| 2 | Final answer is correct, yet reasoning has notable gaps,
unjustified leaps, or mathematical errors.          |
| 3 | Final answer is correct and reasoning is complete, logically
sound, and properly justified throughout.    |

Style Penalty
-------------
- If the writing is flashy or ornate and mathematically
incoherent, subtract 1 point from the rubric score (minimum 0).
- Do not penalize students who explore a wrong path first but
later correct themselves.

Please do not do very long thinking for this task, as else you
will reach the output limit, keep it short strictly.

Output Format (strict)
----------------------
Reply with nothing except a single tag in this exact form (you can
have your chain of thought before coming to the final score in the
below specified format):
<score>n</score>
where where n can be only from the set {0, 1, 2, 3}.
Do not output anything else.
```