
Eliciting Reasoning in LLMs using Logprob-based Rewards

Group Name: RL Barcelona

Github Repo (code): <https://github.com/karaca-i/eliciting-reasoning-LLM-logprob-rewards>,

Supervisor: Auguste Poiroux

Ibrahim Karaca
ibrahim.karaca@epfl.ch

Orhun Ege Celik
orhun.celik@epfl.ch

Semih Zaman
semih.zaman@epfl.ch

Arkadiusz Rozmarynowicz
arkadiusz.rozmarynowicz@epfl.ch

David Grishchuk
david.grishchuk@epfl.ch

Abstract

We explore reinforcement learning (RL) techniques to elicit reasoning capabilities in large language models (LLMs), focusing on log-probability-based rewards as an alternative to rule-based supervision. Using the Qwen3B-Instruct model, we compare models fine-tuned with exact-match rules to those trained using log-likelihood rewards computed as the probability of the correct answer given the generated reasoning trace. To ensure training stability, we apply reward scaling alongside Group Relative Policy Optimization (GRPO). Our experiments on the GSM8K dataset show that, while rule-based training achieves strong results (up to 75% accuracy), logprob-based methods can surpass it when they are appropriately scaled. Specifically, the logprob-scaled-3 model reaches a final accuracy of 79% after 1500 steps, demonstrating the effectiveness of probabilistic, model-intrinsic rewards. These findings underscore the potential of logprob-based reward formulations as scalable and general-purpose alternatives for reasoning in LLMs.

1 Introduction

Recent efforts to align large language models (LLMs) with human reasoning have turned to reinforcement learning (RL) as a tool for eliciting structured, interpretable outputs [1], [24]. While supervised fine-tuning (SFT) remains a standard approach for instruction-following, it often fails to induce step-by-step reasoning without extensive annotation [5]–[7]. In contrast, works like DeepSeek-R1 [5], [20] have shown that RL with structured reward functions can effectively guide models toward logical and verifiable outputs, especially in mathematical problem solving [20].

Most prior approaches use *rule-based rewards*—binary, handcrafted signals that enforce specific structural patterns (e.g., XML tags or exact matches). While effective in narrow domains, such rewards lack generality and fail to capture model uncertainty [16], [18]. Alternatively, *log-probability-based rewards*, computed as the model’s likelihood of the correct answer conditioned on its own reasoning, offer a continuous and model-intrinsic training signal that generalizes across tasks [11], [18], [26].

In this work, we extend the DeepSeek-R1 pipeline and compare two RL strategies for training the Qwen2.5-3B-Instruct model:

- A **rule-based GRPO trainer** using a composite reward over format compliance and correctness.
- A **logprob-based GRPO trainer** using the model’s average per-token log-likelihood of the correct answer.

Both methods are evaluated on the GSM8K dataset using Group Relative Policy Optimization (GRPO), a critic-free RL algorithm. To stabilize logprob training, we apply reward scaling and normalization to mitigate sparsity and sequence-length bias [2].

Contributions

- We implement a log-probability-based reward formulation for reasoning tasks that replaces discrete, rule-based correctness checks with a continuous likelihood-based signal conditioned on model-generated reasoning traces.
- We scale and normalize the log-probability rewards to stabilize training under Group Relative Policy Optimization (GRPO) and compare their effectiveness against traditional rule-based rewards on the GSM8K benchmark.
- We demonstrate that, with appropriate reward scaling, logprob-based rewards can not only match but surpass rule-based performance by achieving 79% accuracy at 1500 steps, while also exhibiting more stable and consistent training dynamics.

The remainder of the report is structured as follows: Section 2 reviews prior work and situates our method in the context of recent reinforcement learning for reasoning. Section 3 outlines our training setup and reward formulations. Section 4 presents experimental results. Section 5 concludes with discussions and future directions.

2 Background and Related Work

The use of reinforcement learning (RL) to improve reasoning in large language models (LLMs) has become a prominent research direction, particularly as the limitations of supervised fine-tuning become increasingly apparent [1], [8], [25]. Traditional supervised learning requires labeled data and can struggle to generalize to open-ended tasks [5], [17]. In contrast, RL offers the flexibility to define reward functions that directly reflect task goals, making it an appealing alternative for tasks where correctness is not binary or supervision is expensive.

DeepSeek-R1 introduced one of the first open-source pipelines to train reasoning-capable LLMs using reinforcement learning [10], [20]. The initial model, DeepSeek-R1-Zero, was trained from scratch using reward functions that evaluated final answer correctness and structural constraints [9], [12]. However, due to stability and readability issues, DeepSeek-R1 introduced a multi-stage process—combining cold-start data, supervised fine-tuning, and GRPO-based RL—to improve robustness. The resulting model achieved 79.8% accuracy on AIME 2024 and 97.3% on MATH-500, closely matching OpenAI’s o1-1217 on reasoning tasks [13].

In a more domain-specific context, DeepSeekMath extended these ideas to mathematical reasoning [14]. The authors pre-trained a model on a 120B-token curated math corpus and then applied GRPO to instruction-tuned checkpoints [14]. Their method led to significant improvements: performance on GSM8K rose from 82.9% (DeepSeekMath-Instruct) to 88.2% (DeepSeekMath-RL), and on MATH from 46.8% (DeepSeekMath-Instruct) to 51.7% (DeepSeekMath-RL) after reinforcement learning [14]. Notably, this was achieved without external toolkits or voting, showcasing the potential of GRPO in purely textual mathematical problem solving. These results highlight how reinforcement learning can elevate model performance to levels comparable to much larger proprietary systems like Minerva 540B [3].

Kimi k1.5 further pushed the boundary by scaling RL to multimodal and long-context settings [15]. Its long-CoT variant, which used length-penalized logprob optimization and online mirror descent, achieved 77.5% on AIME 2024, 96.2% on MATH-500, and 94th percentile on Codeforces, rivaling state-of-the-art models like GPT-4o and Claude 3.5. Even the short-CoT version, trained with distilled long-CoT signals, surpassed baseline models like Qwen2.5 and LLaMA-3 in several benchmarks [15]. These results emphasize that strong reasoning can be acquired through relatively simple RL frameworks, provided sufficient scaling and regularization are applied.

Prior approaches rely on handcrafted, binary rule-based rewards that are task-specific and limited in generality. We instead adopt log-probability-based rewards, which provide a continuous, model-intrinsic signal based on the likelihood of the correct answer given the model’s reasoning [4]. Combined with reward scaling and GRPO, this approach offers broader applicability and smoother training dynamics while matching or exceeding rule-based performance [19].

3 Approach

Our aim is to reproduce the DeepSeek-R1 recipe on a small scale, training a 3-B parameter model for 1000 GRPO steps with two different reward families (rule-based and log-probability) under identical hyper-parameters [13], [20].

3.1 Problem formulation

For a GSM8K prompt p , the policy first samples a reasoning trace r and a prediction \hat{a} . Let a denote the ground-truth answer [2]. GPROTrainer of Unsloth library maximizes:

$$J(\theta) = \mathbb{E}_{p \sim \mathcal{D}} \mathbb{E}_{r, \hat{a} \sim \pi_\theta} [R(p, r, \hat{a}, a)]. \quad (1)$$

where p is prompt sampled from the GSM8K dataset \mathcal{D} , r is reasoning trace, \hat{a} is prediction and a is ground truth [21], [23].

Reward families.

1. **Rule-based.** Five deterministic scorers inspect the XML layout, reasoning and answer tag formats, require an integer answer, and give a binary correctness check in range (0, 2) [22].
2. **Log-probability.** We drop \hat{a} , append the ground truth a to the context, and reward

$$R = \log P(a \mid p, r) \quad (2)$$

Back-propagation normally follows the sampled sequence (p, r, \hat{a}) , which are prompt, reasoning trace and prediction respectively. For the rule-based case that is fine because $\hat{a} = a$ whenever the reward is non-zero. For the log-prob case, however, $\hat{a} \neq a$ in most steps, so gradients would drift toward *wrong* answers. We therefore subclass GRPOTrainer and re-inject the ground truth answer, forcing gradients through (p, r, a) [21].¹

3.2 Google Colab environment, limits, and computation cost

All training runs were executed in a free Google Colab session equipped with a single NVIDIA T4 GPU (15 GB VRAM) and a strict 12-hour wall-clock limit. Generating *eight* 1024-token completions per prompt in 4-bit LoRA mode leaves less than 1 GB of head-room, so out-of-memory stops and idle time-outs forced us to checkpoint every 100 steps and reload the adapter. A single forward-backward pass of GRPO with $G=8$ completions takes ≈ 12 s. Consequently, 1000 steps—our fixed budget for *both* the rule-based and log-prob runs—require ≈ 3.5 h. Due to the tight memory, we used 4-bit quantization, rank-64 LoRA, and capped completions at 200 new tokens, which shortens the reasoning traces but makes the experiments feasible under Colab constraints.

3.3 Reward functions

The two branches differ only in how the scalar reward R is computed for each of the $G = 8$ completions produced from the same prompt.

¹In principle one could back-prop only through p, r to avoid overfitting the answer tokens; however, this is out of scope of this project.

Rule-based reward

Component	Condition checked	Range
XMLCount	Four correctly paired XML tags; $-0.001 \times (\text{length of any tail text})$	$[0, 0.50]$
SoftFormat	Loose regex finds both tags in order anywhere in output.	$[0, 0.50]$
StrictFormat	Strict regex enforces exact newline template.	$[0, 0.50]$
IntAnswer	$\hat{a}.\text{isdigit}() \Rightarrow 0.5$; else 0.	$[0, 0.50]$ or $[0, 1.50]$
Correctness	2.0 if extracted answer equals gold answer.	$[0, 2.00]$

How the five scores are used: For each of the $G = 8$ completions produced from the same prompt we *sum* the five component scores, yielding a raw reward in $[0, 4]$ or $[0, 5]$ based on the *IntAnswer* choice.² We then compute the group-centred advantage $A_g = R_g - \frac{1}{8} \sum_{j=1}^8 R_j$ before applying the GRPO loss [20], [21]. This averaging step removes prompt-level bias while keeping the shaping signal of the individual components.

Log-probability reward

1. **Tag sanity-check:** If the generated text lacks a well-formed `<reasoning>` or `<answer>` block we return $R = -\infty$ so the sample is discarded [22].
2. **Build two strings:** context $c = p + r$, $f = c + \text{<answer>} a \text{</answer>}$.
3. **One forward pass:** Tokenize f , feed it through the model *once*, obtain a logits tensor.
4. **Slice answer logits:** Keep only the rows in ℓ that predict the gold-answer tokens; this avoids storing a $\text{length-}|c|$ softmax we never use.
5. **Selective log-softmax:** Apply the memory-friendly helper `selective_log_softmax` to obtain per-token log-probs $\log P_\theta(a_t | a_{<t}, p, r)$ without materialising the full softmax [4].
6. **Length-normalized mean:** $\tilde{R} = \frac{1}{T} \sum_{t=1}^T \log P_\theta(a_t | \cdot)$ where $T = |a_g|$.
7. **Exponential scaling to $[0, \lambda]$:** The code returns $R = \lambda e^{\tilde{R}}$, bringing typical values into a reasonable $0-\lambda$ range as the rule-based composite.³ (Empirically $\tilde{R} \in [-\infty, 0]$, so $e^{\tilde{R}} \in [0, 1]$ leads to a rule-based-like reward range) [11], [18].

3.4 Group Relative Policy Optimization

Each mini-batch contains G completions per prompt, and for each completion i we compute:

$$A_i = R_i - \frac{1}{G} \sum_{j=1}^G R_j, \quad (3)$$

then minimize the clipped loss with $\varepsilon = 0.2$ and β chosen such that $\text{KL} \approx 0.02$.

$$\mathcal{L}_{\text{GRPO}} = -\mathbb{E} \left[\min(r_t A_t, \text{clip}(r_t, 1 - \varepsilon, 1 + \varepsilon) A_t) \right] + \beta \text{KL}[\pi_{\theta_{\text{old}}} \| \pi_\theta], \quad (4)$$

3.5 Implementation specifics

- **Model:** 4-bit LoRA (unsloth), rank 64, 1024-token context, gradient-checkpointing enabled for each 100 steps.
- **GRPO-Sampler:** 8 completions per prompt, temperature 0.8, top- p 0.95, max 200 generated tokens.
- **Optimizer:** ADAMW-8BIT, 5×10^{-6} learning rate, cosine schedule, grad-norm 0.1.
- **Stability:** reward rescaling to $[0, 3]$ and $[0, 5]$ (log-prob), normalized advantages, and selective log-softmax for memory-efficient rewards [26].

²Maximum $3.5 + 0.5 = 4$ or $3.5 + 1.5 = 5$.

³ λ is set to 2, 3 and 5 as reasonable values to influence the exponential scaling in the subsequent sections.

3.6 Algorithms

Algorithm 1 GRPO with rule-based rewards (N = 1000 steps)

Require: Prompt set \mathcal{D} , policy π_θ , step count $N = 1000$

- 1: **for** $t = 1$ to N **do**
- 2: Sample a batch $\{p^{(k)}\}_{k=1}^B \sim \mathcal{D}$
- 3: **for all** prompts $p^{(k)}$ in the batch **do**
- 4: Generate $G = 8$ completions $\{r_g^{(k)}, \hat{a}_g^{(k)}\}_{g=1}^8$
- 5: **for** $g = 1$ to 8 **do**
- 6: Compute rule-based reward $R_g^{(k)}$ using:
- 7: XMLCount + SoftFormat + StrictFormat + IntAnswer + Correctness
- 8: **end for**
- 9: Compute centred advantages $A_g^{(k)}$ via Eq. (3)
- 10: **end for**
- 11: Update π_θ using GRPO loss from Eq. (4)
- 12: **end for**

Algorithm 2 GRPO with log-probability reward (N = 1000 steps)

Require: Prompt set \mathcal{D} , policy π_θ , step count $N = 1000$

- 1: **for** $t = 1$ to N **do**
- 2: Sample a batch $\{p^{(k)}\}_{k=1}^B \sim \mathcal{D}$
- 3: **for all** prompts $p^{(k)}$ in the batch **do**
- 4: Generate $G = 8$ traces $\{r_g^{(k)}, \hat{a}_g^{(k)}\}_{g=1}^8$
- 5: **for** $g = 1$ to 8 **do**
- 6: Replace $\hat{a}_g^{(k)} \leftarrow a^{(k)}$ (gold answer)
- 7: Compute reward $R_g^{(k)}$ using Eq. (2)
- 8: **end for**
- 9: Compute centred advantages $A_g^{(k)}$ via Eq. (3)
- 10: **end for**
- 11: Update π_θ using GRPO loss from Eq. (4)
- 12: **end for**

4 Results

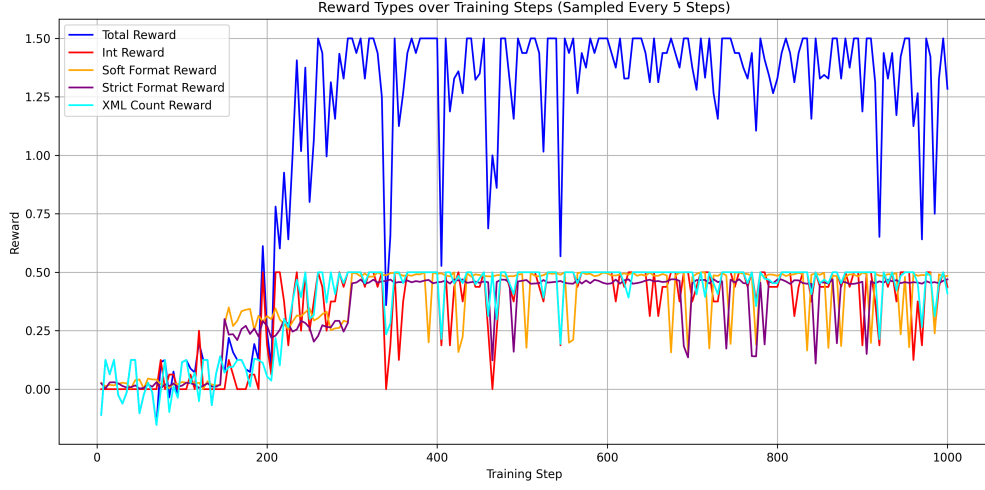
As we tested different approaches, we tracked the accuracy and collected rewards of the results. We then compared effectiveness of our agents.

4.1 Reward Evolution

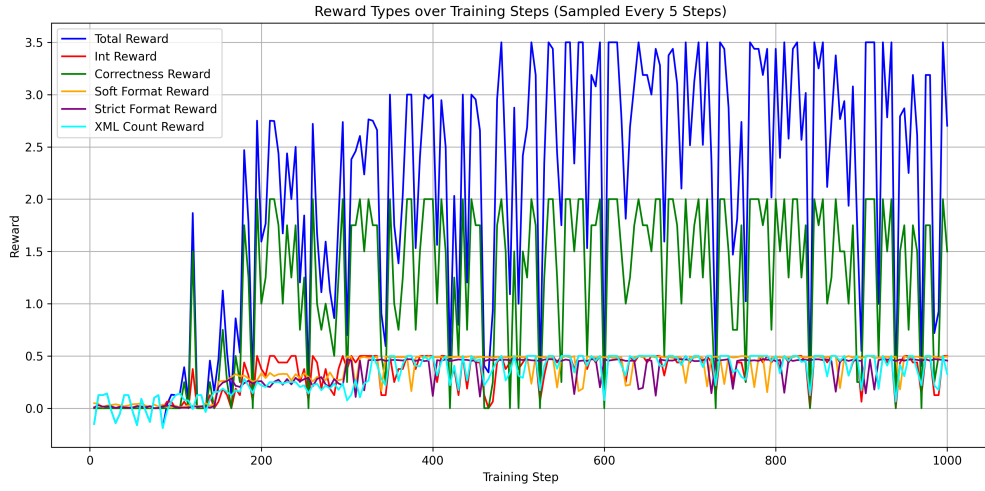
The baseline model demonstrates a rapid increase in total reward during the early phase of training, particularly between steps around 200. This increase is primarily driven by improvements in the soft format reward and XML count reward, both of which indicate that the model quickly learns to produce structurally valid outputs. However, int reward remains comparatively not that much high throughout training, suggesting that the model struggles to generate integer prompts.

By introducing correctness reward, the rule-based model achieves higher total reward compared to baseline model, peaking around 3.5. This improvement is accompanied by a substantial increase in integer reward after steps around 150-200, indicating a stronger alignment between the generated content and the desired numerical structure. While the format-related rewards remain high, they exhibit significant fluctuation, which may reflect instability or overfitting to specific patterns observed in training.

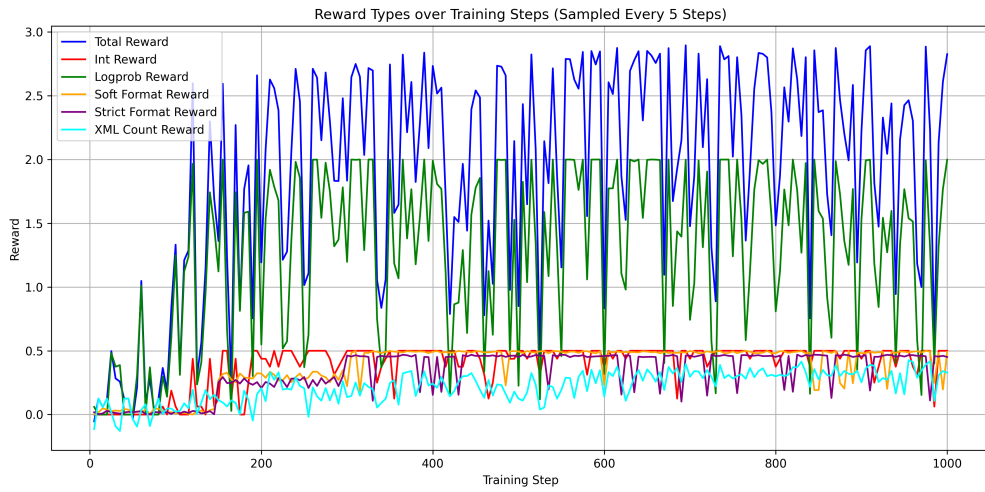
The log-probability-based model exhibits a distinct learning dynamic compared to the other approaches. While its total reward remains lower than that of the rule-based model, it achieves a more stable and consistent improvement in int reward. This indicates that the model is effectively learning to produce outputs that align with the intended meaning. Importantly, the format-related rewards, such as soft format compliance and XML structure correctness, remain comparable to those observed in the baseline and rule-based models, indicating that the model preserves syntactic



(a) Average rewards obtained by our baseline (a reasoning model) over number of steps



(b) Average rewards obtained by the rule-based approach over number of steps



(c) Average rewards obtained by log-prob approach

Figure 1: Reward trajectories for (a) baseline, (b) rule-based, and (c) log-probability models, showing differences in semantic alignment and format compliance during training

fidelity while emphasizing semantic alignment. Overall, the log-probability model favors semantically faithful outputs over rigid structural adherence, and does so with relatively low reward variance, suggesting improved training stability.

4.2 Accuracy Comparison

This section presents a comparison of model accuracies under different reward strategies. We observe a strong correlation between model performance and the reward values provided by the *IntAnswer* function. Our initial comparison uses a reward value of 0.5 for *IntAnswer*.

Figure 3 presents the accuracy of three training strategies: log-probability-based rewards (with two scaling values), exact match-based rewards, and a formatting-only baseline. Accuracy is measured as the number of correct answers over 200 GSM8K math problems. Figure 2 shows an example prediction. We assess the impact of reward scaling by testing log-probability models with scales 2.0 and 5.0, evaluated every 100 steps. The Qwen3B-Instruct model trained with only formatting and *IntAnswer* at 0.5 achieved 25% accuracy. Exact match increased it to 71%, while log-prob rewards reached 67% and 69%. This suggests exact-match rewards outperform log-prob rewards in 1000 steps, though higher scaling helps log-prob models by emphasizing correct formatting.

When the *IntAnswer* reward was increased to 1.5 and the training extended to 1500 steps, the model trained with log-prob rewards began to outperform the rule-based exact match rewards, as shown in Figure 4. The highest accuracies achieved with log-prob rewards scaled by 3 and 5 were 79% and 76% respectively, while the rule-based accuracy remained at 75%. This highlights the potential of log-prob-based rewards, when an appropriate scaling factor is chosen with extended training process.

Question: The Doubtfire sisters are driving home with 7 kittens adopted from the local animal shelter when their mother calls to inform them that their two house cats have just had kittens. She says that Patchy, the first cat, has had thrice the number of adopted kittens, while Trixie, the other cat, has had 12. How many kittens does the Doubtfire family now have?

Response: <reasoning>
 The Doubtfire sisters adopted 7 kittens from the local animal shelter. Patchy has had thrice the number of adopted kittens, which means Patchy had $7 \times 3 = 21$ kittens. Trixie, in turn, had 12 kittens. To find the total number of kittens the Doubtfire family now has, we add the adopted kittens to the kittens from Patchy and Trixie: $7 + 21 + 12 = 40$ kittens.
 </reasoning>
 <answer> 40 </answer>

Figure 2: A question from the GSM8K dataset and the answer of our logprob-based rewarded model.

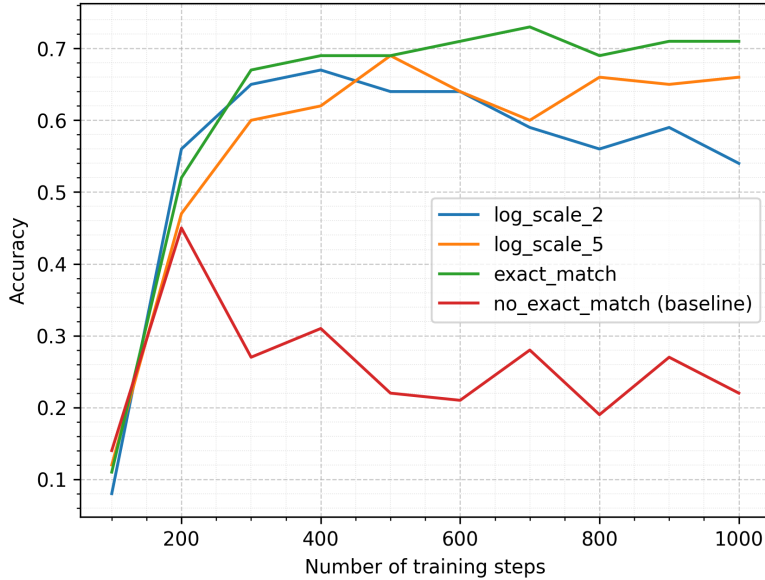


Figure 3: Accuracy comparison of reasoning models trained with different reward functions, using an *IntAnswer* reward value of 0.5.

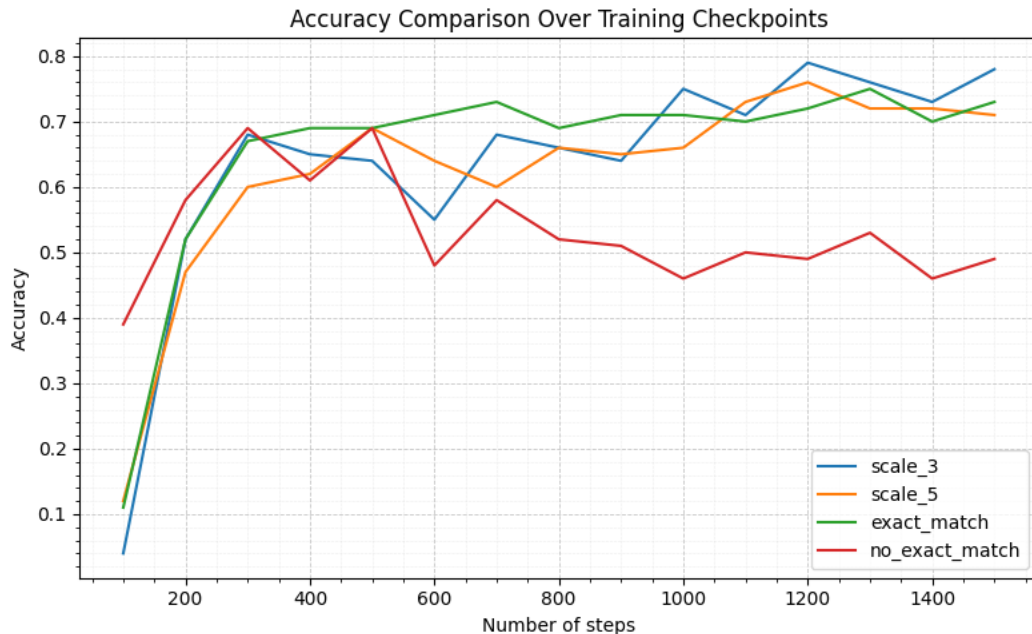


Figure 4: Accuracy comparison of reasoning models trained with different reward functions, using an *IntAnswer* reward value of 1.5.

5 Conclusion

This work compared rule-based and log-probability-based reward strategies for training reasoning-capable LLMs via reinforcement learning. While rule-based supervision achieved strong results, our experiments show that log-probability rewards can match or even exceed its performance when they are appropriately scaled. Notably, the model trained with logprob rewards scaled by 3 reached the highest accuracy of 79% after 1500 steps, outperforming the rule-based model’s 75%. Beyond accuracy, logprob-based methods yielded smoother training dynamics and maintained structural correctness, demonstrating that intrinsic, probabilistic signals can serve as effective alternatives to handcrafted rules. These findings highlight the promise of scalable, general-purpose reward formulations for reasoning in language models. Nevertheless, identifying the optimal scaling factor for logprob-based rewards remains an open question and a promising direction for future work. Training with log-probability-based rewards appears to outperform rule-based rewards over extended training. However, due to the computational constraints of this project, a deeper investigation into how log-prob rewards continue to enhance model accuracy in the long term (with more training steps) is left as future work.

References

- [1] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” *arXiv preprint arXiv:1706.03741*, 2017. [Online]. Available: <https://arxiv.org/abs/1706.03741>.
- [2] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, *et al.*, “Training verifiers to solve math word problems,” *arXiv preprint arXiv:2110.14168*, 2021. [Online]. Available: <https://arxiv.org/abs/2110.14168>.
- [3] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, *et al.*, “Measuring mathematical problem solving with the math dataset,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021. [Online]. Available: <https://arxiv.org/abs/2103.03874>.
- [4] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, *et al.*, “Solving quantitative reasoning problems with language models,” *arXiv preprint arXiv:2206.14858*, 2022. [Online]. Available: <https://arxiv.org/abs/2206.14858>.
- [5] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, *et al.*, “Training language models to follow instructions with human feedback,” *arXiv preprint arXiv:2203.02155*, 2022. [Online]. Available: <https://arxiv.org/abs/2203.02155>.
- [6] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. H. Chi, and D. Zhou, “Self-consistency improves chain-of-thought reasoning in large language models,” *arXiv preprint arXiv:2203.11171*, 2022. [Online]. Available: <https://arxiv.org/abs/2203.11171>.
- [7] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022. [Online]. Available: <https://arxiv.org/abs/2201.11903>.
- [8] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, “Direct preference optimization: Your language model is secretly a reward model,” in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023. [Online]. Available: <https://arxiv.org/abs/2305.18290>.
- [9] HKUST NLP Group, *Simplerl: Lightweight reasoning rl*, GitHub Repository, 2024. [Online]. Available: <https://github.com/hkust-nlp/simpleRL-reason> (visited on 05/29/2025).
- [10] HuggingFace Team, *Open-r1: Reasoning llms*, GitHub Repository, 2024. [Online]. Available: <https://github.com/huggingface/open-r1> (visited on 05/29/2025).
- [11] F. Jiao, G. Guo, X. Zhang, N. F. Chen, S. Joty, and F. Wei, “Preference optimization for reasoning with pseudo feedback,” *arXiv preprint arXiv:2411.16345*, 2024. [Online]. Available: <https://arxiv.org/abs/2411.16345>.
- [12] J. Pan, *Tinyzero: Lightweight reasoning agents*, GitHub Repository, 2024. [Online]. Available: <https://github.com/Jiayi-Pan/TinyZero> (visited on 05/29/2025).
- [13] P. Schmid, *Reproducing deepseek-r1 with 3b models*, Blog Post, 2024. [Online]. Available: <https://www.philschmid.de/mini-deepseek-r1> (visited on 05/29/2025).
- [14] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, *et al.*, “Deepseekmath: Pushing the limits of mathematical reasoning in open language models,” *arXiv preprint arXiv:2402.03300*, 2024. [Online]. Available: <https://arxiv.org/abs/2402.03300>.
- [15] K. Team, “Kimi k1.5: Scaling reinforcement learning with llms,” *arXiv preprint arXiv:2501.12599*, 2024. [Online]. Available: <https://arxiv.org/abs/2501.12599>.
- [16] P. Wang, L. Li, Z. Shao, *et al.*, “Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Association for Computational Linguistics, 2024, pp. 9426–9439.
- [17] Z. Xi, W. Chen, B. Hong, *et al.*, “Training large language models for reasoning through reverse curriculum reinforcement learning,” in *Proceedings of the 41st International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 235, PMLR, 2024, pp. 54 030–54 048.
- [18] G. Cui, L. Yuan, Z. Wang, H. Wang, W. Li, *et al.*, “Process reinforcement through implicit rewards,” *arXiv preprint arXiv:2502.01456*, 2025. [Online]. Available: <https://arxiv.org/abs/2502.01456>.

- [19] S. Dou, M. Wu, J. Xu, *et al.*, “Improving reinforcement learning exploration for LLM reasoning through retrospective replay,” *arXiv preprint arXiv:2504.14363*, 2025.
- [20] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, *et al.*, “Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning,” *arXiv preprint arXiv:2501.12948*, 2025. [Online]. Available: <https://arxiv.org/abs/2501.12948>.
- [21] Unsloth Team, *Advanced grpo fine-tuning for llama 3 (3b)*, Colab Notebook, 2025. [Online]. Available: [https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/Advanced_Llama3_2_\(3B\)_GRPO_LoRA.ipynb](https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/Advanced_Llama3_2_(3B)_GRPO_LoRA.ipynb) (visited on 05/29/2025).
- [22] Unsloth Team, *Gsm8k reward functions*, Documentation, 2025. [Online]. Available: <https://docs.unsloth.ai/basics/reasoning-grpo-and-r1#gsm8k-reward-functions> (visited on 05/29/2025).
- [23] Unsloth Team, *Qwen3 (4b) grpo training*, Colab Notebook, 2025. [Online]. Available: [https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/Qwen3_\(4B\)-GRPO.ipynb](https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/Qwen3_(4B)-GRPO.ipynb) (visited on 05/29/2025).
- [24] Y. Wang, Q. Yang, Z. Zeng, *et al.*, “Reinforcement learning for reasoning in large language models with one training example,” *arXiv preprint arXiv:2504.20571*, 2025.
- [25] R. Xie, D. Qiu, D. Gopinath, *et al.*, “Interleaved reasoning for large language models via reinforcement learning,” *arXiv preprint arXiv:2505.19640*, 2025.
- [26] W. Xiong, H. Zhang, C. Ye, L. Chen, N. Jiang, and T. Zhang, “Self-rewarding correction for mathematical reasoning,” *arXiv preprint arXiv:2502.19613*, 2025. [Online]. Available: <https://arxiv.org/abs/2502.19613>.