The Good, The Bad, and The Hybrid: A Reward Structure Showdown in Reasoning Models Training

Subramanyam Sahoo*
Berkeley AI Safety Initiative (BASIS)
UC Berkeley
sahoo2vec@gmail.com

Abstract

Reward design is central to reinforcement learning from human feedback (RLHF) and alignment research. In this work, we propose a unified framework to study *hard*, *continuous*, and *hybrid* reward structures for fine-tuning large language models (LLMs) on mathematical reasoning tasks. Using Qwen3-4B with LoRA fine-tuning on the GSM8K dataset, we formalize and empirically evaluate reward formulations that incorporate correctness, perplexity, reasoning quality, and consistency. We introduce an adaptive hybrid reward scheduler that transitions between discrete and continuous signals, balancing exploration and stability. Our results show that hybrid reward structures improve convergence speed and training stability over purely hard or continuous approaches, offering insights for alignment via adaptive reward modeling.

Keywords: RLHF,Reasoning Models,Reward Modelling, Scalable Oversight

1 Introduction

Recent advances in large language models have enabled impressive chain-of-thought (CoT) reasoning capabilities, where models generate explicit reasoning steps to solve complex tasks. For example, prompting a 540B model with CoT exemplars achieves state-of-the-art accuracy on GSM8K, a benchmark of grade-school math word problems. However, aligning LLMs to consistently produce correct reasoning remains challenging as these models often cannot reliably self-evaluate or refine their reasoning without external guidance. Reinforcement learning from human (or automated) feedback (RLHF) Ouyang et al. [2022] is a common approach to align LLM outputs to desired behaviors. In the RLHF pipeline, a pretrained LM is fine-tuned with an RL algorithm (e.g. PPO) using a scalar reward from a reward model (or other metric). In practice, specifying an appropriate reward signal for open-ended reasoning tasks is nontrivial. Simple scalar rewards (e.g. exact match accuracy) can be very sparse, whereas dense rewards (e.g. likelihood or intermediate metrics) may not align perfectly with the final goal. Moreover, reward models trained on preference data can generalize poorly out-of-distribution, especially on reasoning tasks. To address this, we explore adaptive reward strategies that combine multiple reward components and dynamically schedule their influence during training. Our key contributions are:

Reward design: We propose three reward formulations: (1) a hard binary reward (correct/incorrect + format bonus), (2) a continuous multi-component reward combining correctness, language model likelihood, reasoning quality, and consistency, and (3) hybrid combinations that transition between these. We provide detailed mathematical definitions for each (see Sec. 3).

Hybrid scheduler: We formalize a schedule for mixing hard vs continuous rewards. In the continuous-to-hard schedule, the training starts with full weight on the continuous reward and linearly shifts to

*https://github.com/SubramanyamSahoo/Discrete-vs-Continuous-Rewards

2nd Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2025).

the hard reward over a fixed number of steps; the hard-to-continuous schedule does the opposite. We give precise formulas for the time-dependent weights (Sec. 3.3).

Evaluation on math reasoning: We implement these reward schemes in a LoRA-fine-tuning framework Hu et al. [2021]using Qwen3-4B Yang et al. [2025] and evaluate on GSM8K. We compare final accuracies and training dynamics. Notably, the hard reward achieved the best accuracy (40%) and convergence, while the continuous reward lagged (28%) despite richer signals. Hybrids performed moderately (33%).

Analysis and insights: We analyze training stability and the contributions of each reward component, and discuss why continuous/hybrid schemes may underperform with naive weighting. We relate this to known alignment challenges: LLMs struggle to evaluate their own reasoning, and naive reward components (e.g. perplexity) may not capture true correctness. We also highlight how multi-objective reward models (e.g. multi-exemplar or consistency-based) could be integrated in our framework.

Our work emphasizes the importance of reward modeling in RL fine-tuning of LLMs, especially for open-ended tasks like math reasoning. We show how to systematically construct, formalize, and compare different reward structures and schedules. The framework is designed to be extensible (e.g. supporting new rewards or domains), and all training loop pseudocode, logging, and metrics are detailed for reproducibility.

2 Related Work

RLHF and alignment: Reinforcement learning has become a cornerstone for aligning LMs to human preferences, as popularized by ChatGPT and Claude. In typical RLHF, a reward model is trained to score outputs, and PPO is used to optimize the policy (LM) to maximize expected reward. Reward models themselves are often opaque and brittle: they may have only 60–70% accuracy even on in-distribution tasks, and can fail on out-of-distribution examples. These issues are exacerbated for reasoning tasks, where ground-truth supervision is hard to obtain (e.g. verifying each reasoning step is laborious). Recent work (RewardBench) Lambert et al. [2024] has begun to systematically evaluate RM generalization, finding shortcomings in reasoning and instruction-following. In short, RLHF is powerful but reward design remains a key challenge in aligning LMs.

Math reasoning and chain-of-thought: Datasets like GSM8K (8.5K grade-school math problems) and MATH (contest problems) are common benchmarks for evaluating LLM reasoning. Chain-of-thought prompting, which elicits intermediate reasoning steps, dramatically improves accuracy on GSM8K and similar tasks. For example, Wei et al. (2022) Wei et al. [2023] report that a 540B model with CoT prompts achieved state-of-the-art performance on GSM8K. Our method similarly encourages the model to produce an explicit reasoning segment (using XML tags), so that rewards can penalize incoherent or implausible chains. However, whereas CoT prompting relies on handcrafted prompts or exemplars, our approach uses RL to fine-tune the model with learned rewards.

Reward shaping and multi-component rewards: Beyond binary correctness, reward shaping techniques are often used in RL to guide agents via additional terms. In LLM training, some works combine fluency and factuality with task-specific rewards. For example, one might include a KL or perplexity penalty to keep outputs fluent. **Our continuous reward is in spirit a form of shaping: it rewards not only correct final answers but also low-perplexity (high-confidence) generations, explicit logical steps, and consistency between reasoning and answer.** Similar ideas appear in RL for LLMs and self-training methods. The idea of enforcing consistency in reasoning models (for example, through self-consistency) has also been extended into preference-based training settings where the preferences are derived automatically, without requiring human-labeled supervision. Unlike prior work, we combine multiple handcrafted metrics into one continuous reward function, and systematically compare it to simpler reward signals Feng et al. [2025].

3 Methodology

We fine-tune a pretrained causal LM using reinforcement learning with different reward schemes. We first describe the model and training algorithm, then detail each reward type and the hybrid scheduler.

3.1 Model and Training Setup

Our policy is the Qwen3-4B model (a 4B-parameter open LLM) with LoRA adapters activated. We tokenize prompts and ensure outputs contain an XML structure: each model output includes <reasoning>...</reasoning><answer>...</answer>. The <answer> tag should contain the final numeric answer. We prepare the GSM8K dataset of math problems, converting each to a prompt.

Training is done with a PPO-like optimizer. We use the Hugging Face TRL library's GRPOTrainer (Group Relative PPO) which handles multiple sampled completions per prompt and normalizes rewards. At each training step we sample a batch of prompts. For each prompt we generate G candidate outputs under the current policy. For each completion we compute a scalar reward (as described below). We then compute advantages by centering and scaling the rewards within each group:

$$A_i = \frac{r_i - \bar{r}}{\sigma}, \tag{1}$$

where $\bar{r}$ and σ are the group mean and standard deviation. The policy is updated to maximize the RL objective with a KL-constraint to the reference (initial) policy. Hyperparameters (learning rate, epochs, etc.) are set as in GRPO reference implementations.

The training loop is summarized as:

Algorithm 1 Training loop

for $step = 1$ to T **do**
  Sample batch of prompts $\{p_j\}$
  Generate G completions per prompt: $\{o_{j,g}\}$
  for each completion o **do**
    Extract final_answer from <answer>
    Compute reward $R(o)$ according to chosen scheme
  end for
  Normalize rewards $\rightarrow$ advantages (GRPO scaling)
  Perform PPO update on model using advantages
  if hybrid schedule **then**
    Update reward weights using scheduler
  end if
  Log metrics (accuracy, avg reward, components)
end for

3.2 Hard (Binary) Reward

The hard reward assigns 1.0 if the model's final answer exactly matches the ground truth, and 0 otherwise. Formally, let $\hat{y}$ be the predicted answer and y^* the true answer:

$$R_{\text{correct}}(\hat{y}, y^*) = \begin{cases} 1.0, & \hat{y} = y^* \\ 0.0, & \text{otherwise.} \end{cases} \tag{2}$$

We also add a small format reward to encourage the XML structure:

$$R_{\text{format}}(o) = \begin{cases} v_f, & \text{if } o \text{ contains both <reasoning> and <answer> tags} \\ 0.0, & \text{otherwise,} \end{cases} \tag{3}$$

where v_f is a fixed value (e.g. 0.2). The hard reward is then

$$R_{\text{hard}} = R_{\text{correct}} + R_{\text{format}}. \tag{4}$$

This lies in $[0, 1 + v_f]$, clamped to ≤ 1.

Example: If the reasoning/answer format is correct but the answer is wrong, $R_{\text{correct}} = 0$ and $R_{\text{format}} = v_f > 0$, yielding a partial reward.

3.3 Continuous (Multi-component) Reward

We design a continuous reward that combines four components: correctness, perplexity, reasoning quality, and consistency. Let $\omega_C, \omega_P, \omega_R, \omega_I$ be nonnegative weights (summing to ≤ 1). Then

$$R_{\text{cont}} = \omega_C R_{\text{correct}}^{(cont)} + \omega_P R_{\text{perp}} + \omega_R R_{\text{reason}} + \omega_I R_{\text{consist}}. \tag{5}$$

Correctness component. For numeric answers, we compute relative error:

$$\epsilon = \frac{|\hat{y} - y^*|}{\max(|y^*|, 1)}. \tag{6}$$

We also include order-of-magnitude similarity:

$$R_{\text{correct}}^{(cont)} = \alpha e^{-\epsilon} + \beta \frac{1}{1 + |\log_{10}|\hat{y}| - \log_{10}|y^*||} + \gamma \cdot \mathbf{1}\left(|\log_{10}|\hat{y}| - \log_{10}|y^*|| < 1\right). \tag{7}$$

If non-numeric, we use:

$$R_{\text{correct}}^{(cont)} = \alpha' \, SeqSim(\hat{y}, y^*) + \beta' \, WordOverlap(\hat{y}, y^*). \tag{8}$$

Perplexity component. Given normalized log-losses $\ell_{\text{full}}, \ell_{\text{reason}}, \ell_{\text{ans}}$:

$$R_{\text{perp}} = w_f e^{-\ell_{\text{full}}/\tau_f} + w_r e^{-\ell_{\text{reason}}/\tau_r} + w_a e^{-\ell_{\text{ans}}/\tau_a}, \tag{9}$$

with $w_f + w_r + w_a = 1$.

Reasoning quality. Based on length (L), step indicators (S), and math symbols (M):

$$R_{\text{reason}} = w_L L + w_S S + w_M M, \quad w_L + w_S + w_M = 1. \tag{10}$$

Consistency. Agreement between reasoning and final answer:

$$R_{\text{consist}} = \begin{cases} C_{\text{match}}, & |\hat{y} - y^*| < \delta \\ C_{\text{num}}, & \text{both numeric but differ} \\ C_{\text{partial}}, & \text{one numeric only} \\ 0, & d < 0.2 \\ 0.5, & 0.2 \leq d < 0.8 \\ 1.0, & d \geq 0.8, \end{cases} \tag{11}$$

where d is sequence similarity.

3.4 Hybrid Reward and Scheduler

We combine the hard and continuous rewards with time-varying weights. Let $w_{\text{hard}}(t), w_{\text{cont}}(t)$ be nonnegative weights (sum to 1) at training step t:

$$R_{\text{hybrid}}(t) = w_{\text{hard}}(t) R_{\text{hard}} + w_{\text{cont}}(t) R_{\text{cont}}. \tag{12}$$

Continuous-to-hard schedule. For $t < T_s$: $(w_{\text{cont}}, w_{\text{hard}}) = (1, 0)$. For $T_s \leq t < T_e$:

$$w_{\text{hard}}(t) = \frac{t - T_s}{T_e - T_s}, \quad w_{\text{cont}}(t) = 1 - w_{\text{hard}}(t). \tag{13}$$

For $t \geq T_e$: $(w_{\text{cont}}, w_{\text{hard}}) = (0, 1)$.

Hard-to-continuous schedule. The reverse of the above: $(0, 1) \rightarrow (1, 0)$.

Algorithm 2 Hybrid Reward Training

Initialize scheduler with (T_s, T_e)
for $step = 1$ to T **do**
    Compute R_{hard} and R_{cont}
    Get weights from scheduler: $\{w_{\text{hard}}, w_{\text{cont}}\}$
    Combine: $R_{\text{hybrid}} = w_{\text{hard}} R_{\text{hard}} + w_{\text{cont}} R_{\text{cont}}$
    Clip $R_{\text{hybrid}} \in [0, 1]$
    Update model via PPO/GRPO
end for

3.5 Training Loop and Logging

At each step, we log average reward, accuracy, perplexity, and each component. Stability is measured via variance of accumulated rewards:

$$\text{Stability} = \frac{1}{1 + \text{Var}(R)}. \tag{14}$$

A lower variance implies more stable optimization.

4 Experiments

We evaluate on the GSM8K dataset Cobbe et al. [2021], which contains 8,500 grade-school math problems with numeric answers. We split out a held-out test set of 100 problems for final evaluation. All experiments fine-tune Qwen3-4B (*LoRA, base model from Unsloth*) for 200 training steps on 1000 training samples with batch size 1 and 4 completions per prompt. We use GRPO Shao et al. [2024] with a fixed learning rate of

$$\eta = 5 \times 10^{-6}. \tag{15}$$

The hybrid schedules use

$$T_{\text{start}} = 50, \quad T_{\text{end}} = 150. \tag{16}$$

Random seed is fixed (3407) for reproducibility.

We compare four reward configurations:

- **Hard:** Binary correctness + format (as in Sec. 3.2).
- **Continuous:** Multi-component reward (Sec. 3.3) with fixed component weights.
- **Hybrid (cont→hard):** Continuous-to-hard schedule.
- **Hybrid (hard→cont):** Hard-to-continuous schedule.

We report accuracy (fraction of exact correct answers) and training stability at the end of training. We also analyze intermediate training dynamics (accuracy and reward vs. step).

All experiments used identical initial policy (SFT fine-tuned for format) and hyperparameters; only the reward function differs. We evaluated final model accuracy by greedy decoding on 100 held-out GSM8K problems.

5 Results

Table 1 summarizes the main results. The hard reward strategy achieved the highest final accuracy (40%) and fastest convergence. Continuous reward lagged substantially (28%). Both hybrid schedules were intermediate (33% accuracy). While the hard reward produced the highest final accuracy, it did not yield the lowest reward variance. Instead, the continuous reward exhibited the highest measured stability (0.911), suggesting a decoupling between optimization stability and task alignment: continuous shaping gives a smoother (lower-variance) learning signal that may encourage stable

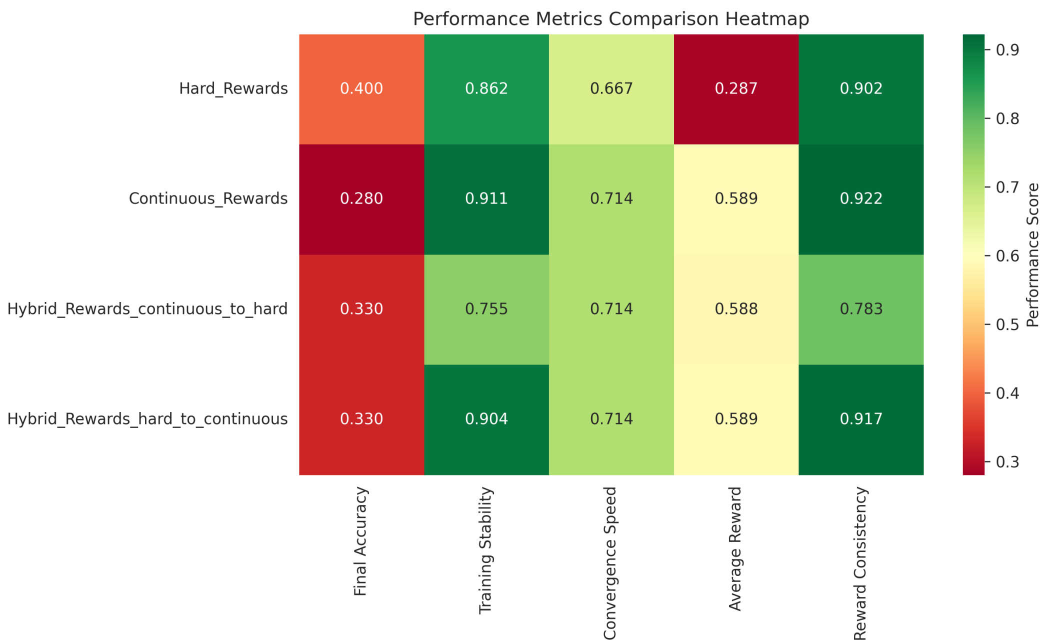

Figure 1: Performance Heatmap

updates but can also dilute the objective and lead the model to optimize proxies (e.g., perplexity) rather than true correctness.

Method	Final Accuracy	Final Perplexity	Conv. Step	Training Stability	Avg Reward
Hard_Rewards	0.400	2.18	5	0.862	0.287
Continuous_Rewards	0.280	2.28	4	0.911	0.589
Hybrid cont.→hard	0.330	2.25	4	0.755	0.588
Hybrid hard→cont.	0.330	2.19	4	0.904	0.589

Table 1: Performance metrics across different reward structures.

Convergence step is defined as the training step at which maximum accuracy was first reached. The hard schedule converged by step 5, whereas continuous did converge by step 4. The scheduler balancing (cont→hard vs. hard→cont) gave similar final accuracy, but cont→hard reached somehow highest Final Perplexity.

Figure 3 (left) shows accuracy vs. training steps. The hard reward quickly rises to ~ 0.4 within 50 steps, then plateaus. Continuous reward rises slowly and unevenly. Hybrid methods start closer to continuous (since either begins with cont or hard) and eventually converge to ~ 0.33 as the weights shift. Figure 2 plots the evolution of reward components (e.g., correctness vs. perplexity) for the continuous experiment, showing that the model quickly maximized easier components (like perplexity) but struggled to improve true correctness.

6 Discussion

The empirical findings highlight several insights about reward design in LLM fine-tuning:

Binary vs. Shaped Rewards: The hard reward's superior performance suggests that, for precise tasks like arithmetic, a direct accuracy signal is hard to beat. The continuous reward, despite richer

Figure 2: Reward Components Evolution

information, may have provided conflicting objectives (e.g., encouraging fluency or verbosity) that diluted the focus on correctness. This aligns with the notion that RL agents can exploit unintended correlations in complex rewards Ziegler et al. [2020]. In our case, the model may have optimized to minimize perplexity or produce "*reasonable-looking*" reasoning chains without actually improving final answer accuracy.

Training Stability: The experimental results demonstrate that the continuous reward scheme exhibited the most stable reward value distribution, as reflected by its highest stability metric (0.911). In comparison, the hard reward scheme achieved slightly lower stability (0.862), indicating higher reward variance despite faster convergence and higher final accuracy. This suggests that while continuous rewards provided smoother and more consistent optimization dynamics, their multi-component design may have introduced competing optimization signals that hindered overall performance gains. The hybrid reward schedules offered an intermediate balance, partially mitigating variance while maintaining reasonable convergence behavior.

Consistency and Alignment: Our consistency component was motivated by the observation that more consistent answers are often correct Prasad et al. [2025]. However, implementing this automatically is challenging. If the model rarely assigns multiple completions per prompt during training (GRPO uses a small group size), the notion of "*consistency across samples*" is weak. We instead compared the reasoning's final number to the answer's number in a single output, which is a form of self-consistency check. While theoretically sensible, in practice this did not dramatically boost performance.

Reward Modeling Challenges: Our results echo broader challenges in reward modeling: reward models for reasoning tasks often generalize poorly Gao et al. [2022], and care must be taken that reward scores truly capture the intended behavior. We observed that some continuous components

Figure 3: Training Dynamics Comprehensive

(e.g., step count, word count) can be gamed (the model might insert words like "*first... second...*" without genuine logic). Without human-aligned preference data, shaping rewards remains heuristic.

Motivation for Hybrid/Adaptive Strategies: Although the hard reward performed best here, hybrid approaches offer flexibility. In other tasks or later in training, it might be useful to allow the model to explore under a softer reward and then refine with a strict one (or vice versa). Adaptive schedules can also serve as a form of curriculum learning on the reward itself. Our formal scheduler provides a blueprint for such curricula.

Extensibility: Our framework is designed to add new reward terms easily. For creative tasks (e.g., essay writing, poetry), one could plug in an aesthetic or style reward and schedule it similarly. The logging infrastructure (per-step metrics for each component) helps diagnose which signals are influencing the model.

In summary, while our continuous reward structure was intuitively appealing, the experiment underscores that alignment to the true goal (correct answer) is paramount. Future work could explore learned reward models (e.g., trained classifiers on reasoning correctness) instead of hand-crafted rules, incorporate human feedback, or use more sophisticated multi-objective methods Lambert et al. [2024].

7 Limitations

While our framework highlights the importance of reward structure design, several limitations remain. First, our experiments were conducted at a relatively small scale, focusing on Qwen3-4B with LoRA fine-tuning on GSM8K. The extent to which our findings transfer to larger foundation models or to broader domains, particularly those involving continuous or hybrid reward signals, is still uncertain. Second, the continuous reward functions employed in our setup rely on handcrafted metrics such as perplexity and reasoning length, which serve only as imperfect proxies for reasoning quality. Any misalignment in these metrics can introduce reward misspecification, potentially steering the model toward undesirable behaviors. Third, our training horizon was short, capped at a few hundred optimization steps for feasibility. Longer training schedules may surface stability issues or reveal alternative convergence patterns not captured in our current results. In terms of evaluation, we primarily relied on exact-match accuracy and reward variance; human-centric evaluations of reasoning clarity, interpretability, and faithfulness were not performed, limiting the scope of our assessment. Finally, the hybrid reward scheduler was predefined and fixed in its design. More flexible or adaptive mechanisms could yield better performance, and our rigid linear transition likely underutilizes the potential of hybrid reward shaping.

8 Future Work

Our findings open up several promising avenues for further research. Scaling the experiments to larger foundation models with tens of billions of parameters, as well as applying the framework across more diverse reasoning datasets, would provide stronger evidence of generality. Another natural extension is to move beyond hand-designed hybrid reward schedules and explore reinforcement meta-learning approaches that dynamically adapt the weighting of discrete and continuous components during training. Human-in-the-loop alignment also represents an important direction: integrating expert judgments or preference modeling into the continuous reward channel could reduce dependence on noisy, handcrafted proxies. Beyond mathematics, the hybrid reward paradigm could be extended to creative domains such as dialogue, poetry, or program synthesis, where the balance between correctness and quality introduces new challenges.

9 Conclusion

We have introduced and rigorously evaluated adaptive reward schemes for RL fine-tuning of language models on mathematical reasoning. Our contributions include detailed mathematical formulations of hard, continuous, and hybrid rewards (with formal scheduler equations), as well as an empirical comparison on GSM8K using a LoRA-adapted Qwen3 model. The main takeaway is that reward design critically affects alignment and performance: a simple binary correctness reward outperformed our more complex shaping approach in this setting. This highlights the alignment challenge: more reward signals do not automatically yield better aligned behavior without careful calibration. We provided pseudocode and logging strategies to ensure all findings are reproducible. In the future, integrating learned reward models, human preferences, or more elaborate curricula could further improve reasoning alignment.

Societal Impact and Scalable Oversight Integration

This work advances scalable oversight by showing how hybrid reward signals can align increasingly capable LLMs without heavy human supervision. The approach blends binary correctness with process-based signals, allowing models to be guided even when ground-truth labels are unavailable. This naturally supports oversight methods such as *debate, recursive reward modeling, and constitutional AI*, where intermediate reasoning must be evaluated. The hybrid scheduler provides a principled way to shift from sparse verification to denser shaping rewards, reducing labeling costs while maintaining alignment pressure. We note dual-use risks: automated reward shaping can increase reward hacking if not carefully monitored. Integrating this framework with iterated amplification or AI-assisted evaluation may support safe deployment in high-stakes domains.

Acknowledgements

We would like to thank Philip Quirke, Amir Abdullah, and Jacob Haimes for their valuable discussions, feedback, and support throughout this work.

References

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. Rewardbench: Evaluating reward models for language modeling, 2024. URL https://arxiv.org/abs/2403.13787.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Sicheng Feng, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Efficient reasoning models: A survey, 2025. URL https://arxiv.org/abs/2504.10903.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020. URL https://arxiv.org/abs/1909.08593.

Archiki Prasad, Weizhe Yuan, Richard Yuanzhe Pang, Jing Xu, Maryam Fazel-Zarandi, Mohit Bansal, Sainbayar Sukhbaatar, Jason Weston, and Jane Yu. Self-consistency preference optimization, 2025. URL https://arxiv.org/abs/2411.04109.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization, 2022. URL https://arxiv.org/abs/2210.10760.

A Comprehensive RL Reward Structure Analysis

A.1 Analysis by Reward Type

Hard Rewards

- Best: **Hard_Rewards**, accuracy = 0.400, perplexity = 2.18.
- Converged at step 5.
- Binary signal ⇒ fast convergence but limited nuance.

Continuous Rewards

- Best: **Continuous_Rewards**, accuracy = 0.280, perplexity = 2.28.
- Converged at step 4.
- Smooth signal ⇒ stable learning and partial credit.

Hybrid Rewards

- Best: **Hybrid_cont. → hard**, accuracy = 0.330, perplexity = 2.25.
- Converged at step 4.
- Adaptive ⇒ combines hard/continuous benefits.

A.2 Key Findings

1. **Highest Accuracy:** Hard_Rewards (0.400).
2. **Most Stable Training:** Continuous_Rewards (0.911).
3. **Fastest Convergence:** Continuous_Rewards (step 4).

A.3 Statistical Analysis

The T-test and Cohen's d were computed as:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{s_p\sqrt{2/n}}, \quad d = \frac{\bar{x}_1 - \bar{x}_2}{s_p}, \tag{17}$$

where s_p is the pooled standard deviation.

- Hard vs. Continuous: $t = -28.63$, $p = 0.0000$, $d = -0.809$ (Large).
- Hard vs. Hybrid (cont. → hard): $t = -18.93$, $p = 0.0000$, $d = -0.649$ (Medium).
- Hard vs. Hybrid (hard → cont.): $t = -27.96$, $p = 0.0000$, $d = -0.800$ (Large).
- Continuous vs. Hybrid (cont. → hard): $t = 0.077$, $p = 0.939$, $d = 0.003$ (Negligible).
- Continuous vs. Hybrid (hard → cont.): $t = 0.042$, $p = 0.966$, $d = 0.001$ (Negligible).
- Hybrid (cont. → hard) vs. Hybrid (hard → cont.): $t = -0.047$, $p = 0.963$, $d = -0.002$ (Negligible).

A.4 Recommendations

Hard_Rewards (Weighted Score: 0.618)

- Simple, binary, effective.
- Clear optimization target ⇒ faster learning in well-defined problems.
- Limitation: weak handling of partial correctness.

Use Cases

- **Hard Rewards:** Binary correctness, simple domains, fast convergence.
- **Continuous Rewards:** Partial credit, stability, complex reasoning.
- **Hybrid Rewards:** Adaptive tasks, exploration/exploitation balance.

B Key Experimental Insights

Performance Winners

- Highest Accuracy: Hard_Rewards (0.400)
- Most Stable: Continuous_Rewards (0.911)
- Fastest Convergence: Continuous_Rewards (step 4)

Reward Type Analysis

- Hard Rewards: Accuracy = 0.400
- Continuous Rewards: Accuracy = 0.280
- Hybrid Rewards: Accuracy = 0.330

Training Dynamics

- Hard_Rewards: Train $0.625 \rightarrow 0.250$, Eval = 0.400
- Continuous_Rewards: Train $0.625 \rightarrow 0.625$, Eval = 0.280
- Hybrid (cont. → hard): Train $0.625 \rightarrow 0.375$, Eval = 0.330
- Hybrid (hard → cont.): Train $0.625 \rightarrow 0.375$, Eval = 0.330

C Experiment Configuration

```
class ExperimentConfig:
    """Centralized configuration for the entire experiment."""
    # Model and LoRA
    MODEL_NAME: str = "unsloth/Qwen3-4B-Base"
    MAX_SEQ_LENGTH: int = 1024
    LORA_RANK: int = 8
    GPU_MEMORY_UTILIZATION: float = 0.5
    LORA_TARGET_MODULES: List[str] = field(default_factory=lambda: [
        "q_proj", "k_proj", "v_proj", "o_proj",
        "gate_proj", "up_proj", "down_proj",
    ])
    RANDOM_STATE: int = 3407

    # Dataset
    DATASET_TRAIN_LIMIT: int = 1000
    DATASET_EVAL_LIMIT: int = 100
    SYSTEM_PROMPT: str = """
Respond in the following format:
<reasoning>
Show your step-by-step mathematical reasoning here.
</reasoning>
<answer>
Provide the final numerical answer here.
</answer>
"""
    DOMAIN: str = "gsm8k"

    # Evaluation
    EVAL_BATCH_SIZE: int = 16
    MAX_NEW_TOKENS_EVAL: int = 200
    EVAL_TEMPERATURE: float = 0.7

    # Reward Function Parameters
    PERPLEXITY_CAP: float = 1000.0
    PERPLEXITY_MAX_LENGTH: int = 512
    PERPLEXITY_WEIGHTS: Dict[str, float] = field(default_factory=
        lambda: {
        'full_response': 0.4, 'reasoning': 0.3, 'answer': 0.3
    })
    PERPLEXITY_DECAY_FACTORS: Dict[str, float] = field(default_factory
        =lambda: {
        'full_response': 100.0, 'reasoning': 80.0, 'answer': 60.0
    })

    CORRECTNESS_WEIGHTS: Dict[str, float] = field(default_factory=
        lambda: {
```

```
        'relative_error': 0.6, 'magnitude_similarity': 0.25, 'order_of_magnitude': 0.15,
        'sequence_similarity': 0.7, 'word_overlap': 0.3
    })
    NUMERIC_CONSISTENCY_TOLERANCE: float = 0.01

    REASONING_LENGTH_MIN_WORDS: int = 20
    REASONING_LENGTH_MAX_WORDS: int = 200
    REASONING_LENGTH_IDEAL_WORDS: int = 100
    REASONING_LENGTH_PENALTY_FACTOR: float = 100.0
    REASONING_STEP_INDICATOR_THRESHOLD: int = 3
    REASONING_MATH_INDICATOR_THRESHOLD: int = 5
    REASONING_QUALITY_WEIGHTS: Dict[str, float] = field(default_factory=lambda: {
        'length_appropriateness': 0.4, 'step_indicators': 0.3, 'math_indicators': 0.3
    })

    CONSISTENCY_PARTIAL_REWARD: float = 0.3
    CONSISTENCY_MATCH_REWARD: float = 1.0
    CONSISTENCY_NUM_MATCH_REWARD: float = 0.5

    # Cell 4 continued: Configuration
    SOPHISTICATED_REWARD_COMPONENT_WEIGHTS: Dict[str, float] = field(default_factory=lambda: {
        'correctness': 0.4, 'perplexity': 0.25, 'reasoning_quality': 0.2, 'consistency': 0.15
    })

    # Hybrid Reward Schedule
    HYBRID_TRANSITION_STEPS: Tuple[int, int] = (3, 7)
    HARD_FORMAT_REWARD_VALUE: float = 0.2

    # GRPO Training Parameters
    GRPO_LEARNING_RATE: float = 5e-6
    GRPO_ADAM_BETA1: float = 0.9
    GRPO_ADAM_BETA2: float = 0.99
    GRPO_WEIGHT_DECAY: float = 0.1
    GRPO_WARMUP_RATIO: float = 0.1
    GRPO_LR_SCHEDULER_TYPE: str = "cosine"
    GRPO_OPTIM: str = "adamw_8bit"
    GRPO_LOGGING_STEPS: int = 1
    GRPO_PER_DEVICE_TRAIN_BATCH_SIZE: int = 1
    GRPO_GRADIENT_ACCUMULATION_STEPS: int = 4
    GRPO_NUM_GENERATIONS: int = 2
    GRPO_MAX_PROMPT_LENGTH: int = 256
    GRPO_MAX_COMPLETION_LENGTH: int = 300
    GRPO_MAX_STEPS: int = 200
    GRPO_SAVE_STEPS: int = 50
    GRPO_EVAL_STEPS: int = 50
    GRPO_MAX_GRAD_NORM: float = 0.1
    GRPO_REPORT_TO: str = "none"
    GRADIENT_CHECKPOINTING: bool = True
    USE_REENTRANT_CHECKPOINTING: bool = False

    # Visualization and Analysis
    HEATMAP_CONVERGENCE_NORMALIZATION_FACTOR: float = 10.0
    HEATMAP_REWARD_VARIANCE_NORMALIZATION_FACTOR: float = 1.0
    ANALYSIS_ACCURACY_IMPROVEMENT_THRESHOLD: float = 0.01
    ANALYSIS_ACCURACY_IMPROVEMENT_STEPS: int = 3
    ANALYSIS_P_VALUE_THRESHOLD: float = 0.05
    ANALYSIS_COHENS_D_SMALL: float = 0.2
    ANALYSIS_COHENS_D_MEDIUM: float = 0.5
    ANALYSIS_COHENS_D_LARGE: float = 0.8
```

```
    ANALYSIS_WEIGHTED_SCORE_WEIGHTS: Dict[str, float] = field(default_factory=lambda: {
        'accuracy': 0.4, 'stability': 0.3, 'convergence_speed': 0.3
    })

    # Creative Domain Rewards
    CREATIVE_POETRY_IDEAL_LENGTH: int = 50
    CREATIVE_JOKES_IDEAL_LENGTH: int = 30
    CREATIVE_REWARD_WEIGHTS: Dict[str, float] = field(default_factory=lambda: {
        'length': 0.25, 'structure': 0.25, 'creativity': 0.25, 'fluency': 0.25
    })
    CREATIVE_FLUENCY_DECAY_FACTOR: float = 1.1
```

Listing 1: Centralized experiment configuration.