Comparative Study of Reward Structures in Reinforcement Learning for Language Models: Hard Labels vs. Perplexity-Based Rewards

Pratham Singla

Abstract

Reinforcement learning (RL) plays a critical role in aligning large language models with task-specific behaviors. This study explores the impact of two different reward structures used during RL fine-tuning: (a) a discrete reward based on correctness of answers and format compliance (as implemented in Unsloth), and (b) a continuous reward based on perplexity reduction or completion likelihood improvement, as proposed in recent literature. I implement both reward schemes and compare their influence on model behavior using key evaluation metrics such as accuracy progression, reward stability, and convergence characteristics. The goal is to understand how each reward type influences training dynamics and performance. The complete implementation and training pipeline are available at: GitHub

1 Methodology

In this project, I fine-tuned a large language model to perform better at mathematical reasoning tasks using reinforcement learning. The core method I used is Group Relative Policy Optimization (GRPO), which iteratively improves the model using the outputs generated by the model itself. The key idea behind GRPO is to increase the likelihood of better (higher-reward) completions, while ensuring that the updated model does not deviate significantly from the initial policy.

1.1 Model and Training Pipeline

I used the Qwen1.5-4B Base model provided by the Unsloth framework as it supports efficient fine-tuning of large models by reducing memory usage and training time without sacrificing performance.

The complete fine-tuning process involved two main steps:

1. **Supervised Fine-Tuning (SFT)**: First, I performed a pre-fine-tuning step where I trained the model using a custom format designed for GRPO reward extraction. This step ensured that model outputs followed a specific structure necessary for computing rewards.

2. **Reinforcement Learning with GRPOTrainer**: After SFT, I trained the model using GRPO with two different reward structures: (a) hard reward signals based on answer correctness and format compliance, and (b) soft rewards based on perplexity improvements. Both training runs were conducted on a subset of a mathematical reasoning dataset. Dataset details are provided in Section 2.

1.2 Hard Reward Structure

In this setup, rewards are given based on whether the model output matches the expected format and the correct final answer. In this three main components are used to compute the reward:

- **Exact Format Matching:** A fixed reward of 3 points is given if the output strictly matches the expected structure using a regular expression. Otherwise, the score is zero.
- **Approximate Format Matching:** Partial rewards (0.5 each) are given if the output includes required tokens such as "`<reasoning_end>`," "`<solution_start>`," and "`<solution_end>`" exactly once. Missing or repeated tokens are penalized.
- **Answer Matching:** This component evaluates the correctness of the predicted final answer:
  - A full reward of 5.0 points is given for an exact match.
  - 3.5 points are given for answers that differ only by whitespace.
  - For numerically close values (within 10% or 20% of the true answer), the model receives 1.5 to 2.0 points.
  - Completely incorrect or unparsable answers result in a negative reward.

These components are combined to compute a total reward per output, which is logged to Weights and Biases (WandB) for real-time monitoring of the training process. The hard reward structure provides clear and direct feedback to the model but may lead to sparse learning signals.

1.3 Perplexity-Based Reward Structure

The second reward strategy I used is based on **perplexity improvement**, as proposed in a recent work [GL25]. This method assigns a reward based on how much the model's generated response helps in predicting the correct reasoning, thereby encouraging the model to produce more useful and informative completions.

1.3.1 What is Perplexity?

Perplexity (PPL) is a standard metric in language modeling used to measure how well a probability model predicts a sample. Lower perplexity implies the model finds the sequence more predictable or likely.

Given a sequence of tokens $y = (y_1, y_2, \ldots, y_T)$, the perplexity is defined as:

$$\text{PPL}(y) = \exp\left(-\frac{1}{T}\sum_{t=1}^{T}\log p(y_t \mid y_{<t})\right)$$

In this project, I compute perplexity in two settings:

- $\textbf{PPL}_\pi(y \mid x)$: Perplexity of the gold reasoning y, given only the question x.
- $\textbf{PPL}_\pi(y \mid x, a)$: Perplexity of y, given the question x and the model-generated answer a.

Here, x is the input question, a is the model's answer, and y is the ground-truth reasoning.

1.3.2 Improvement Score

The improvement is measured by how much the perplexity of the gold reasoning improves when conditioned on the model's answer. It is computed as:

$$I_\pi(x, y, a) = \left(1 - \frac{\text{PPL}_\pi(y \mid x, a)}{\text{PPL}_\pi(y \mid x)}\right) \times 100$$

A positive value of I_π indicates that the answer a helps reduce the perplexity of y, which is desirable.

1.3.3 Bounded Reward Function

To convert the improvement score into a usable reward signal, I used a bounded reward function as defined in the original paper. The reward R is assigned based on the value of I_π as follows:

$$R(x,a,y) = \begin{cases} 0, & I(x,a,y) < 0.05 \\ 0.5, & 0.05 \leq I(x,a,y) < 1 \\ 0.9, & 1 \leq I(SI_i, c_{i+1}, \hat{p}) < 2 \\ 1, & I(x,a,y) \geq 2 \end{cases}$$

This mapping ensures that rewards are:

- Zero if the answer is not helpful.
- Gradually increased as the answer improves the model's ability to predict the gold reasoning.
- Limited to a maximum of 1 to keep the values stable.

1.3.4 Final Reward Calculation

I combined the bounded reward from perplexity improvement with the format-based reward functions used in the hard reward setup. This hybrid reward setup provides continuous learning feedback while maintaining the structural integrity of outputs.

All reward components, perplexity metrics, and batch-level summaries were logged using Weights and Biases (WandB) for analysis and visualization.

2 Dataset

This project uses two datasets during the fine-tuning process: one for supervised fine-tuning (SFT) and another for reinforcement learning (GRPO training). Both datasets are focused on mathematical reasoning tasks.

2.1 Supervised Fine-Tuning (SFT)

For the SFT stage, I used a filtered subset of NVIDIA's Open Math Reasoning dataset [MHS+25]. The original dataset includes a wide range of mathematical problems and model traces. I specifically selected only the high-quality `DeepSeek R1` traces, which are known for their well-structured reasoning paths. This filtering step was already supported and pre-processed within the Unsloth training notebook.

2.2 GRPO Fine-Tuning

For the reinforcement learning phase using GRPO, I needed a dataset that included ground truth reasoning steps (referred to as `gold_reasoning`) in addition to the questions and final answers. This was necessary to compute the perplexity improvement for each model output.

To fulfill this requirement, I used a custom subset of a math reasoning dataset [ZWP+25], which is available publicly at: HuggingFace

Due to computational constraints, I limited the dataset to:

- **50 training samples** for fine-tuning the model with GRPO.
- **200 test samples** used for evaluating final accuracy and perplexity improvements.

Each sample in the dataset contains:

- A math question.
- The expected final answer.
- A detailed step-by-step explanation (gold reasoning).

This structure allowed the model to be evaluated both in terms of output accuracy and how helpful its responses were in predicting the correct reasoning.

3 Results

I was able to successfully complete the reinforcement learning fine-tuning process using the hard label reward structure. The final fine-tuned model can be accessed at: HuggingFace

Due to memory constraints, I could not complete training using the perplexity-based soft label reward. Even after switching to a smaller model, `unsloth/gemma-3-1b-it-unsloth-bnb-4bit`, and reducing the number of generations and batch size, the out-of-memory issue persisted. The complete code for this implementation is available in the project repository.

3.1 Performance with Hard Label Rewards

Final Test Accuracy: Using the test set of 200 samples, the model achieved a final answer accuracy of **59.5%**, which demonstrates moderate performance considering the limited number of training examples and compute resources used.

3.1.1 Training Analysis

Figure 1 shows the variation in the improvement rate across global training steps. The fluctuations indicate instability in the model's ability to consistently generate better completions during training. While there are several positive spikes which suggests that the model improved on certain batches but struggled to maintain consistent progress, likely due to the discrete nature of hard rewards.

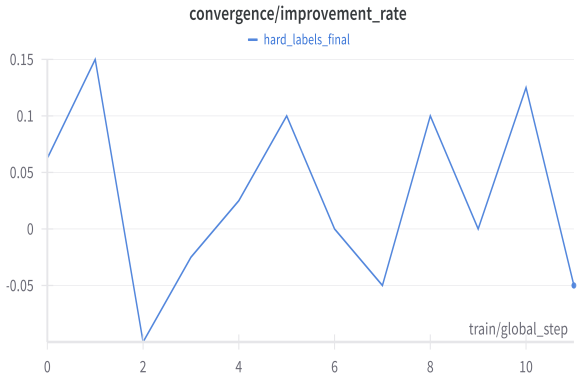

Figure 1: Improvement rate during GRPO training with hard labels.

In Figure 2, the trend in average reward values is initially decreasing after which it begins to improve steadily. This pattern implies that the model initially struggles to meet the strict hard label criteria but eventually begins to learn representations that yield higher rewards.

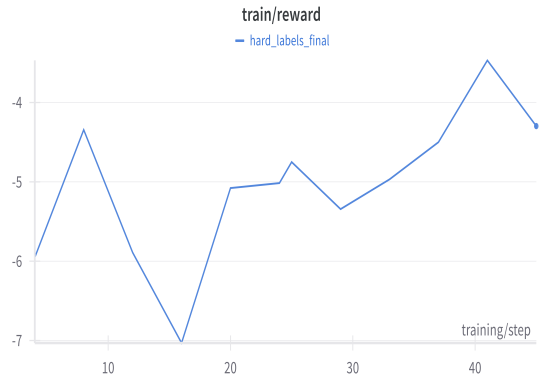

Figure 2: Average reward during training using hard label rewards.

Figure 3: Training loss during GRPO fine-tuning with hard labels.

3.2 Hypotheses on Perplexity-Based Soft Labels

While I could not complete training with the soft-label reward structure, several hypotheses can be made based on its design and partial trials:

- Convergence might be smoother, due to the continuous nature of the reward signal.
- Accuracy could improve, as the reward is based on how much the answer helps the model predict the correct reasoning.
- Since the reward structure is softer and based on the model's probability space, it may lead to more stable gradients and prevent sharp oscillations in learning dynamics.

However, the effectiveness of soft reward training likely depends on the domain and structure of the task. In structured reasoning tasks like math, where correctness is binary and highly structured, soft rewards may not always be superior.

4 Discussions

4.1 Why did I pick this project?

I had previously worked on a research project that explored how aware LLMs are of their own learned reasoning policies. That study highlighted how GRPO-trained models often generalize well but sometimes show weak alignment between internal reasoning and final answers. This raised an interesting question: would using a continuous reward function, such as perplexity-based improvement, help bridge this gap and lead to more aligned and interpretable reasoning? It seemed like a natural follow-up experiment and aligned well with my broader interest in LLMs and reinforcement learning.

4.2 If I had more compute/time, what would I have done?

With more compute, I would have completed the training using the soft-label reward structure, overcoming the out-of-memory issues. This would allow a direct comparison between hard and soft reward paradigms based on real results rather than hypotheses. I would also test the approach on larger and more diverse datasets, or even across different domains, to better understand its generalizability.

4.3 What did I learn in this project?

This project helped me understand how to implement custom reward functions in the GRPO training loop, especially when combining multiple evaluation criteria like output format and answer quality. I also gained practical experience in debugging tokenization/model embedding mismatches that lead to CUDA trigger errors. Additionally, I learned the importance of managing GPU memory efficiently during reinforcement learning with language models.

4.4 What surprised me the most?

The biggest surprise was how unstable the training process can be when using strict reward rules. Even though hard labels are simple, they still enabled decent learning with just 50 training samples

4.5 If I had to write a paper on this project, what else would need to be done?

To turn this project into a formal paper, I would:

- Extend the comparison to include other RL baselines such as PPO and DPO to contextualize GRPO's behavior under different reward schemes.
- Explore how these reward structures generalize to domains where rewards are not strictly verifiable—such as open-ended text generation (e.g., storytelling, poetry, or humor).

References

[GL25] Alexander Gurung and Mirella Lapata. Learning to reason for long-form story generation, 2025.

[MHS+25] Ivan Moshkov, Darragh Hanley, Ivan Sorokin, Shubham Toshniwal, Christof Henkel, Benedikt Schifferer, Wei Du, and Igor Gitman. Aimo-2 winning solution: Building state-of-the-art mathematical reasoning models with openmathreasoning dataset, 2025.

[ZWP+25] Han Zhao, Haotian Wang, Yiping Peng, Sitong Zhao, Xiaoyu Tian, Shuaiting Chen, Yunjie Ji, and Xiangang Li. 1.4 million open-source distilled reasoning dataset to empower large language model training, 2025.