Summary of Research at UIUC

Author: Zhizheng Zhao

🎯 1. Introduction

This document summarizes a series of research explorations aimed at optimizing the Group-Based Reward Policy Optimization (GRPO) framework for Large Language Models. The original GRPO method, while effective, presents several potential weaknesses.

My research was structured into two primary investigations:

- **Enhancing Advantage Granularity:** Moving from the coarse, sample-level advantage to a more fine-grained, token-level advantage.
- **Addressing Distribution Mismatch:** Tackling the "hard prompt" problem, where the model's sampling distribution fails to overlap with the reward distribution, leading to training stagnation.

📊 Experiment Overview (Summary Table)

Experiment	Focus Area	Core Idea / Hypothesis	Result
Exp 1	Token-Level Granularity	Critic learns token "vitality"	🔴 **Failed**
Exp 2	Token-Level Granularity	"Seize rare opportunities"	🔴 **Failed**
Exp 3	Distribution Mismatch	"Concentration of force"	🟡 **No Change**
Exp 4	Distribution Mismatch	Suppress known "bad" regions	🟠 **Inconclusive** (Missed Opp.)
Exp 5	Distribution Mismatch	Reasoning quality $\propto$ GT Likelihood	🔴 **Failed**

🔬 2. Part 1: Enhancing Advantage Granularity

Motivation: The primary dissatisfaction was with GRPO's sample-level advantage A_{sample}. This single scalar value is applied uniformly to all tokens, failing to capture the intuition that some tokens in a high-reward sample are more "vital" or "critical" than others.

2.1 Experiment 1: Token-Level Advantage Bias via Critic Model 🤖

Methodology

- **Core Idea (Hypothesis):** A critic model can be trained to assign an "advantage bias" to each token. The critic's training dynamic should naturally reward "good" tokens. If a token consistently appears in high-advantage samples ($A_{sample} > 0$), the loss function will push its bias $b_\phi(t)$ higher. Conversely, a "bad" token consistently appearing in low-advantage samples ($A_{sample} < 0$) will have its bias pushed lower. This creates a learned, fine-grained credit assignment.
- **Implementation:**
  1. A model with a modified "head" is used as the critic (parameters ϕ).
  2. For a given sample, the critic outputs a bias $b_\phi(t)$ for each token t.
  3. The final per-token advantage is $A_{token}(t) = A_{sample} + b_\phi(t)$.
  4. The critic is trained *concurrently* with the policy, using the following loss function:

$$L(\phi) = -A_{sample} \cdot \sum_{t \in \text{sample}} b_\phi(t) \tag{1}$$

Results

This approach was unsuccessful. The training process proved to be highly unstable. The added complexity of training a second, large critic model did not yield any performance gains and, in most cases, failed to converge or beat the original GRPO baseline.

2.2 Experiment 2: Advantage Compensation for Probability Fluctuations 📈

Methodology

- **Core Idea (Hypothesis):** We must "seize rare opportunities." A high-quality sample ($A_{sample} > 0$) that was *unlikely* to be generated (i.e., it contains very low-probability tokens) is an extremely valuable learning signal. A single, standard gradient update (from $L_{PG} = -A_{sample} \cdot \sum_t \log p(t)$) provides an **insufficient probability boost**. Because of the low-probability tokens, this valuable sample is **unlikely to be sampled again** in future steps, and the learning opportunity is lost. Therefore, we must amplify the advantage *immediately* to force the model to learn aggressively from these "rare gems" in a single step, accelerating convergence.
- **Implementation:**
  1. For a positive advantage sample ($A_{sample} > 0$), find and p_{max} (max token probabilities within that sample).
  2. Calculate a compensation factor C.

$$C = 2 - \frac{p_t}{p_{max}} \tag{2}$$

  3. Modify the advantage used for the loss calculation:

$$A_{token}(t) = A_{sample} \cdot C = A_{sample} \cdot \left(2 - \frac{p_t}{p_{max}}\right) \tag{3}$$

(Example: If p_t is half of p_{max}, $C = 1.5$, amplifying the advantage by 50%.)

Results

This approach also failed. The method suffered from severe training instability. The amplified advantage, especially in early training, led to large, erratic gradient updates that prevented the policy from converging. It consistently performed worse than the GRPO baseline.

🗺️ 3. Part 2: Addressing Sampling & Reward Distribution Mismatch

Motivation: After the token-level explorations, my focus shifted to a more fundamental GRPO problem: the mismatch between the model's **sampling distribution** (the answers it *generates*) and the **reward distribution** (the answers that are *correct*).

> **The Core Problem (The "All-Fail" Group):**
> For difficult prompts, the model's initial policy has zero overlap with the high-reward region. All N samples in a group receive a reward of 0. This results in $A_{sample} = 0$, meaning **no gradient is computed, and no learning occurs** for this prompt. This creates a vicious cycle:
>
> - **Epoch 1:** Sample $\rightarrow$ All Fail $\rightarrow A = 0 \rightarrow$ No Gradient
> - **Epoch 2:** Policy is unchanged $\rightarrow$ Sample $\rightarrow$ All Fail $\rightarrow A = 0$
>
> This "hard subset" of the training data is effectively ignored.

3.1 Experiment 3: Computation Re-allocation by Culling Solved Prompts ✂️

Methodology

- **Core Idea (Hypothesis):** A "concentration of force" approach. By culling "solved" prompts (where all N samples are correct) from the training set, we stop wasting computational resources on them. This re-allocates the *entire* sampling and gradient budget to the remaining, unsolved prompts, especially the "all-fail" groups. The goal was to break through the "all-fail" barrier with the *same* computational cost, or achieve the same performance in less time.
- **Implementation:**
  1. **Identify "Solved" Prompts:** Track prompts where all N samples receive the maximum reward.
  2. **Cull from Next Epoch:** Remove these solved prompts from the training set in the *next* epoch.
  3. **Re-allocate Resources:** The model now only trains on the difficult subset.

Results

This method showed no significant change. The training curves were nearly identical to the baseline. The intuition gained was that breaking the "all-fail" barrier requires an enormous increase in sampling (perhaps $> 10\times$), and this simple re-allocation was insufficient.

3.2 Experiment 4: Negative Advantage for All-Failure Groups 📉

Methodology

- **Core Idea (Hypothesis):** If the model's high-probability sampling region *consistently* yields zero reward, that region is "bad." We should explicitly suppress it. The problem is that the "all-fail" group produces $A = 0$, leading to $\nabla L = 0$. By setting $A = -c$ (a small negative constant, e.g., -1), we change the loss. The total probability of all actions must sum to 1. By decreasing the probability of this known "bad" region, we *necessarily* increase the probability of *other*, unexplored actions, hopefully allowing the model to discover the reward region.
- **Implementation:**
    1. If a group consists entirely of 0-reward samples, manually override $A_{sample} = 0$ to $A_{sample} = -c$.
    2. The policy loss $L = -A_{sample} \cdot \sum \log p(t)$ becomes:
        - **Standard Case:** $L = -(0) \cdot \sum \log p(t) = 0$. (No gradient)
        - **Proposed Case:** $L = -(-c) \cdot \sum \log p(t) = c \cdot \sum \log p(t)$.
    3. Gradient *descent* on this new positive loss will *decrease* the log-probabilities of these failed samples.

Results

This method showed minor, but not significant, improvement. The uplift was not substantial enough to be conclusive, given the small models and limited epoch counts.

> *(Personal Note: I abandoned this approach prematurely. Subsequent research from other teams has since validated this exact idea as a viable method for improving exploration in sparse-reward settings, which was a missed opportunity. [NGRPO: Negative-enhanced Group Relative Policy Optimization]().)*

3.3 Experiment 5: Ground Truth Likelihood as a Dense Reward 💡

Methodology

- **Core Idea (Hypothesis):** This was targeted at math reasoning tasks. The hypothesis is that **a good reasoning chain should make the correct answer more probable, *even if the model generates the wrong one*.** Instead of a binary $R \in \{0, 1\}$, we can create a dense reward. We assume that $R_{new} = P_\theta(\text{GT Answer} \mid \text{Model's Reasoning})$ is a smooth signal where better reasoning (even if it leads to a wrong final number) gets a higher reward than nonsensical reasoning.
- **Implementation:**
    1. For an "all-fail" group, take each failed sample.
    2. Decode the text. Replace the model's wrong answer (e.g., `## Answer: 123`) with the known correct answer (e.g., `## Answer: 456`).
    3. Calculate the likelihood of *only the ground truth answer*, conditioned on the model's *original reasoning*.

$$R_{new} = P_\theta(\text{\#\# Answer: 456} \mid \text{Model's Original Reasoning}) \tag{4}$$

    4. This `likelihood` R_{new} is now used as the reward, replacing $R = 0$.

Results

This method failed. While it successfully "solved" the $A = 0$ problem by providing a dense reward, the quality of the training signal was extremely poor.

- **Analysis:** The method creates a **fundamental logical disconnect**. It trains the model to associate a line of reasoning (e.g., "Step 1: 2+2=4. Step 2: 4*3=12.") with a ground truth answer that does not follow from it (e.g., "Answer: 7"). This "incoherent" training data confused the model, and performance degraded.

🏁 4. Conclusion

The series of explorations yielded significant insights, primarily in the form of **"what *doesn't* work."**

- **Token-level advantages** (via critics or heuristics) are highly prone to instability and add complexity that is not justified by performance.
- The **"all-fail" group problem** is a genuine and critical flaw in GRPO for hard tasks. My attempts (culling, negative advantage, GT-likelihood) correctly identified the core challenge: it's a fundamental exploration problem that requires a more direct intervention on the gradient or reward signal.