# A Minimalist Approach to LLM Reasoning: from Rejection Sampling to Reinforce

Wei Xiong Jiarui Yao Yuhui Xu Bo Pang Lei Wang
Doyen Sahoo Junnan Li Nan Jiang Tong Zhang Caiming Xiong

Presenter: Yicheng Tao

July 4, 2025

# Outline

# Outline

# Background and Related Works

## Background

- Reinforcement learning algorithms have been widely used in post-training of LLMs for **mathematical reasoning** tasks.
- **GRPO** stands out for its success in training DeepSeek-R1, though it lacks a comprehensive justification of the algorithmic advantages.
- **RAFT** is one of the simplest and most interpretable baselines showing good empirical performance.

# Background and Related Works

## Data filtering in LLM Post-Training

- Discard candidates except for the top and bottom-ranked responses to reduce noise.
- Remove prompts that are too easy or too hard.
- Filter out responses with incorrect answers (RAFT).

## LLM for Mathematical Reasoning

- Syntactic dataset and supervised fine-tuning.
- RL with verifier-based rewards.
- Complex reasoning strategies (Backward search, self-correction, etc.)

# Overview

## Revisiting RAFT and GRPO

- RAFT trains solely on positive samples, leading to a rapid reduction in policy entropy and eventually being surpassed by GRPO.
- GRPO implicitly filters out harmful prompts with all-negative responses, which contributes to most of the performance gain.

## Reinforce-Rej

Motivated by the study with RAFT and Reinforce, a new Reinforce variant, ***Reinforce-Rej***, which selectively filters out prompts with either all correct or all incorrect responses, is proposed. This method enjoys comparable final performance to GRPO, and demonstrates superior KL efficiency.

# Outline

# RAFT: Reward-ranked fine-tuning

## Data Collection

For a batch of prompts $\{x_1, x_2, \ldots, x_M\}$, sample $n$ responses $\{a_{i,1}, a_{i,2}, \ldots, a_{i,n}\}$ for each $x_i$ from the LLM.

## Rejection Sampling

Let $r(x, a) \in \{-1, 1\}$ be the binary reward function. Compute the reward for each response of $x_i$ as $r_{i,j}, j = 1, 2, \ldots, n$. Retain only the responses with $r_{i,j} = 1$ to form a dataset $\mathcal{D}$.

## Model Fine-Tuning

Let $\pi$ be the current policy. Maximize the log-likelihood over the selected dataset:

$$\mathcal{L}^{\mathsf{RAFT}}(\theta) = \sum_{(x,a)\in\mathcal{D}} \log \pi_\theta(a|x)$$

# Policy Gradient and Reinforce

## Learning Objective

Maximize the expectation of rewards gained by the policy model $\pi_\theta$.

$$J(\theta) = \mathbb{E}_{x\sim d_0}\left[\mathbb{E}_{a\sim\pi_\theta(\cdot|x)} r(x,a)\right] \tag{1}$$

With replay buffer and importance sampling:

$$J(\theta) = \mathbb{E}_{x\sim d_0}\left[\mathbb{E}_{a\sim\pi_{\theta_{\text{old}}}(\cdot|x)}\left[\frac{\pi_\theta(a|x)}{\pi_{\theta_{\text{old}}}(a|x)} r(x,a)\right]\right]. \tag{2}$$

With clipping techniques from PPO:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|}\sum_{x,a\in\mathcal{D}}\left[\min\left(\frac{\pi_\theta(a|x)}{\pi_{\theta_{\text{old}}}(a|x)} r(x,a), \text{clip}\left(\frac{\pi_\theta(a|x)}{\pi_{\theta_{\text{old}}}(a|x)}, 1-\epsilon, 1+\epsilon\right)\cdot r(x,a)\right)\right] \tag{3}$$

# Loss function for autoregressive models

Let $a$ be a response from LLM, and $\{a_1, a_2, \ldots, a_n\}$ are the tokens.

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{x,a \in \mathcal{D}} \frac{1}{|a|} \sum_{t=1}^{|a|} \left[ \min\left( s_t(\theta) \cdot r(x,a), \mathsf{clip}(s_t(\theta), 1-\epsilon, 1+\epsilon) \cdot r(x,a) \right) \right], \quad (4)$$

where $s_t(\theta) = \frac{\pi_\theta(a_t|x, a_{1:t-1})}{\pi_{\theta_{\mathsf{old}}}(a_t|x, a_{1:t-1})}$.

# GRPO

The loss function of GRPO is similar to (4), with the reward $r(x, a)$replaced by advantage function $A_t(x, a)$ for the $t$-th token. For each prompt $x$, GRPO will sample $n$ responses and compute the following advantage for the $t$-th token of the $i$-th response:

$$A_t(x, a_i) = \frac{r_i - \mathsf{mean}(r_1, \ldots, r_n)}{\mathsf{std}(r_1, \ldots, r_n)}.$$

This normalization serves to reduce the variance of the stochastic gradient.

# Outline

# RAFT++ and Reinforce-Rej

## RAFT++

By adding an indicator function $\mathcal{I}\left(r(x,a) = \arg\max_i r(x,a_i)\right)$ to (4), we can obtain the loss function of RAFT++.

The indicator ensures that we only train on the response with the highest reward (positive samples).

## Reinforce-Rej

The loss function of Reinforce-Rej is the same as (4), while the dataset is constructed by removing the prompts with either all correct or all incorrect responses.

# Summary

Table 1: Comparison of the tricks used in different algorithms.

| | Importance Sampling | Clipping | Reject Sampling | Advantage |
|---|---|---|---|---|
| RAFT | × | × | ✓ | × |
| RAFT++ | ✓ | ✓ | ✓ | × |
| Reinforce | ✓ | ✓ | × | × |
| GRPO | ✓ | ✓ | × | ✓ |
| Reinforce-Rej | ✓ | ✓ | ✓ | × |

- RAFT++ rejects all negative samples.
- Reinforce-Rej rejects all prompts with either all correct or all incorrect responses.

# Outline

# Experiment Setup

## Dataset and Models

- Numina-Math: 860k math problems with labeled ground-truth answers.
- Qwen2.5-Math-7B-base and LLaMA-3.2-3B-instruct.

## Evaluation

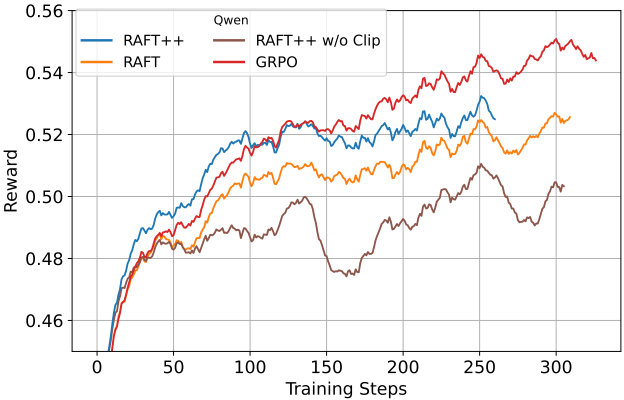
- Benchmark: Math500, Minerva Math, Olympiad Bench.
- Use average@16 for evaluation.
- AIME2024 only contains 30 problems and the trend is noisy for all algorithms.

# Main Result

| Model | Algorithm | Math500 | Minerva Math | Olympiad Bench | Average |
|---|---|---|---|---|---|
| Qwen2.5-Math-7B-base | Base | 41.3 | 11.0 | 18.6 | 23.6 |
| | RAFT | 77.4 | 40.8 | 38.6 | 52.3 |
| | RAFT++ | 80.2 | 44.9 | 43.3 | 56.1 |
| | Iterative DPO | 76.0 | 31.2 | 39.3 | 48.8 |
| | Reinforce | 80.1 | 40.7 | 40.9 | 53.9 |
| | GRPO | 81.3 | 45.5 | 42.2 | 56.3 |
| | PPO | 79.0 | 39.3 | 39.1 | 52.5 |
| | Reinforce-Rej | 81.9 | 44.2 | 43.1 | 56.4 |
| LLaMA-3.2-3B-instruct | Base | 26.3 | 7.4 | 5.5 | 13.1 |
| | RAFT | 46.1 | 17.6 | 13.9 | 25.9 |
| | RAFT++ | 47.4 | 19.1 | 16.3 | 27.6 |
| | Reinforce | 45.9 | 13.7 | 13.0 | 24.2 |
| | GRPO | 49.2 | 19.3 | 16.8 | 28.4 |
| | PPO | 46.5 | 19 | 15.1 | 26.9 |
| | Reinforce-Rej | 50.1 | 19.3 | 16.1 | 28.5 |

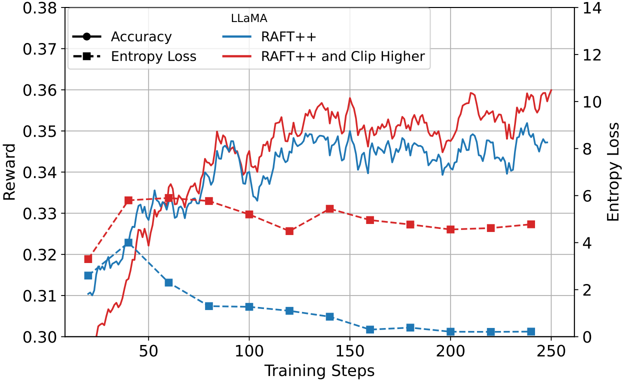
- RAFT and RAFT++ approach deep RL methods with surprisingly small performance gap.

# Effects of Distribution Correction and Clipping



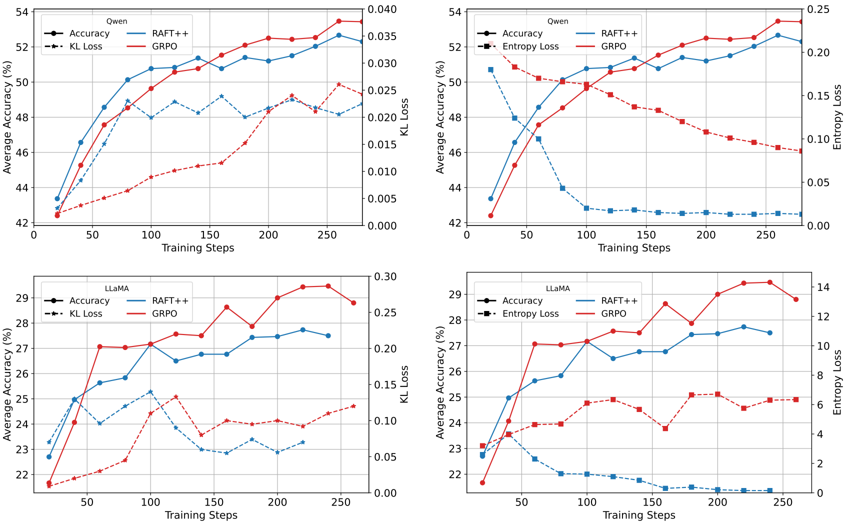

- Although clipping may be infrequent, unbounded updates can lead to instability and degraded performance.
- RAFT++ achieves faster early-stage convergence but is surpassed by GRPO.

# Effects of Clipping Higher

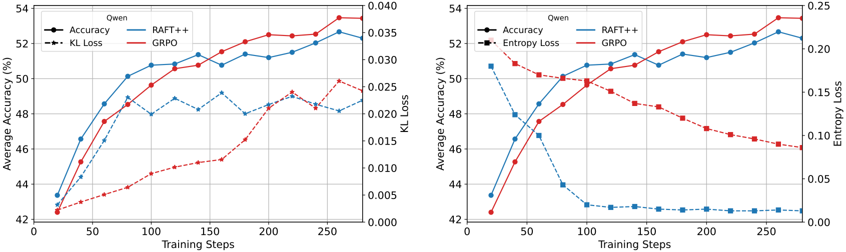



- Clipping higher leads to better performance with stable entropy loss.

# Effects of Reject Sampling

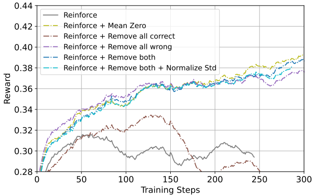


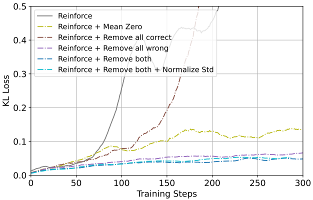
# Effects of Reject Sampling

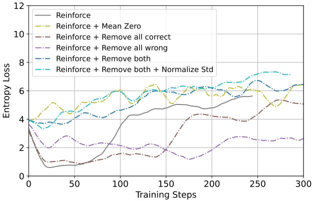



- Learning from only positive samples leads to faster convergence and entropy collapse.
- Stable policy entropy enhances performance, which matches the observation in higher clipping experiment.

# Effects of Reject Sampling

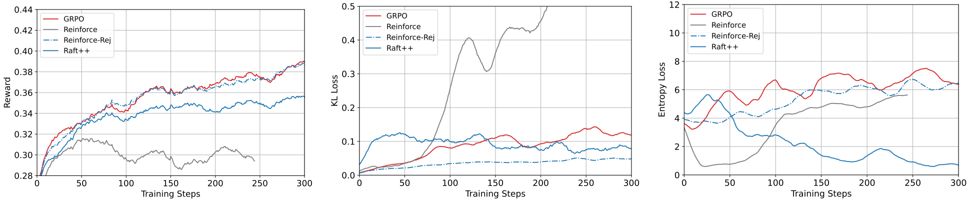



- Removing both all negative and all positive samples leads to better performance.
- Variance normalization is not a key contributor to performance.
- "Reinforce + Mean Zero" variant shows increased KL loss and does not improve rewards, indicating potential instability.

# Effects of Reject Sampling




- The core strength of GRPO lies in rejecting low-quality (especially incorrect) samples, rather than normalization.

# Outline

# Conclusion

- The utility of negative samples in RL-based LLM training is nuanced.
  - Rejecting all negative samples leads to faster convergence and entropy collapse.
  - Harmful prompts with all incorrect and all correct responses will degrade performance.
- The success of GRPO is not due to variance normalization, but rather the implicit filtering of negative samples.
- By adopting rejection sampling strategies, RAFT++ and Reinforce-Rej can serve as lightweight, interpretable, and effective baselines for future work on reward-driven LLM post-training.