Comparing GRPO and PPO on Traditional Reinforcement Learning Tasks

Tong Shen
Department of Applied Mathematics and statistics
Stony Brook University
Stony Brook, NY 11794
tong.shen.1@stonybrook.edu

Huifang Xiang
Department of Computer Engineering
Stony Brook University
Stony Brook, NY 11794
huifang.xiang@stonybrook.edu

Guangying Wang
Department of Electrical Engineering
Stony Brook University
Stony Brook, NY 11794
guangying.wang@stonybrook.edu

Abstract

In this project, we compare the performance of two famous Reinforcement Learning algorithms, Proximal Policy Optimization (PPO) and Group Relative Policy Optimization (GRPO) on traditional Atari game Galaxian. We show that although GRPO works well for mathematical reasoning in large language models scenarios, it fails to perform effectively on traditional reinforcement learning tasks. Our experiments and analysis show that the absence of a pretrained reference model is the key reason GRPO underperforms in vision-based RL environments.

1 Introduction

Reinforcement Learning (RL) has become a cornerstone of modern Large Language Model (LLM) training. While early application of RL in LLMs focused preventing harmful output via techniques like RLHF (Reinforcement Learning from Human Feedback), the current frontier has shifted towards enhancing complex reasoning capabilities, particularly in mathematical reasoning and theorem proof. Unlike creative writing, mathematical reasoning requires strict logical consistency and multi-step verification, tasks where standard supervised fine-tuning (SFT) often falls short. To bridge this gap, RL algorithms are employed to incentivize self-correction and chain-of-thought generation. Among these algorithms, Proximal Policy Optimization (PPO) and Group Relative Policy Optimization (GRPO) are both widely used in RL phase. GRPO, in particular, was introduced to improve mathematical reasoning and has shown notable efficiency advantages over PPO in aligning LLMs.

However, there is a lack of systematic studies comparing PPO and GRPO on traditional RL environments such as Atari games, which form the foundation of modern RL benchmarks. In this project, we aim to implement and compare these two algorithms on classic RL tasks to examine whether the efficiency gains observed in LLM alignment transfer to these domains. We hypothesize that GRPO may exhibit reduced performance in traditional RL tasks due to the absence of a powerful reference model, which plays a central role in its success in LLM training.

Our specific objective is to apply PPO and GRPO to the classic Atari game *Galaxian*. This environment features high-dimensional visual input, discrete action space, delayed rewards, and long-term sequential decision-making. The goal is to train an agent that maximizes its score by effectively shooting enemy targets while avoiding being hit, thereby demonstrating competent strategic play and survival behavior.

Preprint. Under review.

The paper is structured as follows. In Sec.2 we review the applications of RL in LLM training, especially on the mathematical reasoning ability of the model. In Sec.3 we review the PPO and GRPO algorithm. Then in Sec.4 we introduce our testbed for traditional Atari games, including the game rules and how to implement using Python. The result is shown in Sec.5, and finally the discussion of the result is in Sec.6.

2 Reinforcement Learning For LLM reasoning

The role of RL in LLM training has evolved through two major stages. In the first stage, RL was primarily used for aligning pretrained models with human perference, using techniques such as Reinforcement Learning from Human Feedback (RLHF). In the second stage, the RL was used to enhance mathematical reasoning and problem solving such as the method RL to incentivize Chain-of-Thought (CoT). One example is shown in Figure 1. Compared with gpt-4o which only use RLHF, gpt-o1 includes the reward signal from the generation of Cot, and this exhibit performance gains across competition-level math, programming, and PhD-level science benchmarks. These results highlight that RL is no longer only an alignment mechanism—rather, it has become a core driver of advanced reasoning capabilities in LLMs.

A notable recent example is DeepSeek-Prover-V1.5, a 7-billion-parameter model that outperforms all open-source systems in formal theorem proving assistant Lean 4, and have comparable ability with powerful closed-source model like GPT and Gemini. The Figure 2 illustrates superiority of DeepSeek-Prover-V1.5 in formal theorem proving: with RL-based optimization, it achieves substantially higher pass rates than all competing systems across both miniF2F and ProofNet benchmarks.

Two vital techniques introduced by Deepseek-Prover-V1.5 is Group Relative Policy Optimization (GRPO) algorithm and Monte-Carlo Tree Search. GRPO is an extension of the famous PPO algorithm, but discards the value model in PPO and use group average to replace the baseline of action-value function. Value model in PPO often has comparable size with the policy model, so GRPO has lower memory and computational burden. Its training pipeline relies on RL: during training, the model receives feedback from a proof assistant and is optimized using the GRPO algorithm, while inference pipeline employs Monte-Carlo Tree Search to navigate and refine proof trajectories. This combination of RL-based optimization and structured search demonstrates how reinforcement learning can fundamentally enhance a model's ability to perform complex, multi-step reasoning. Importantly, GRPO is also suitable for the comparative nature of rewards models, since reward models are typically trained on comparisons of outputs of the same prompt.

However, this success invites a deeper investigation. GRPO works well in tasks like mathematical reasoning because the pretrained model already produces candidate proofs, enabling reliable group-wise comparison. Whether GRPO can achieve similar success or still beats PPo in traditional reinforcement learning environments—such as Atari games—where there is no powerful pretrained reference model is still a open question.

To explore this problem, our project compares PPO and GRPO on the Atari Galaxian task. Our specific objective is to apply PPO and GRPO to the classic Atari game Galaxian. This environment has high-dimensional visual input, discrete action space, delayed rewards, and long-term sequential decision-making. The goal is to train an agent that maximizes its score by effectively shooting enemy targets while avoiding being hit, thereby demonstrating competent strategic play and survival behavior.

3 Algorithms

In this section, we briefly go through the design and development of PPO and GRPO. We will first introduce the policy gradient and TRPO algorithm, which as the fundamental of the two algorithms we interested in.

3.1 Policy Gradient Methods

Policy Gradient methods are a class of Reinforcement Learning algorithms that directly optimize the policy parameters θ. The core objective is to identify an optimal policy π_θ that maximizes the cumulative expected return. Distinct from Value-based methods, PG methods update parameters by

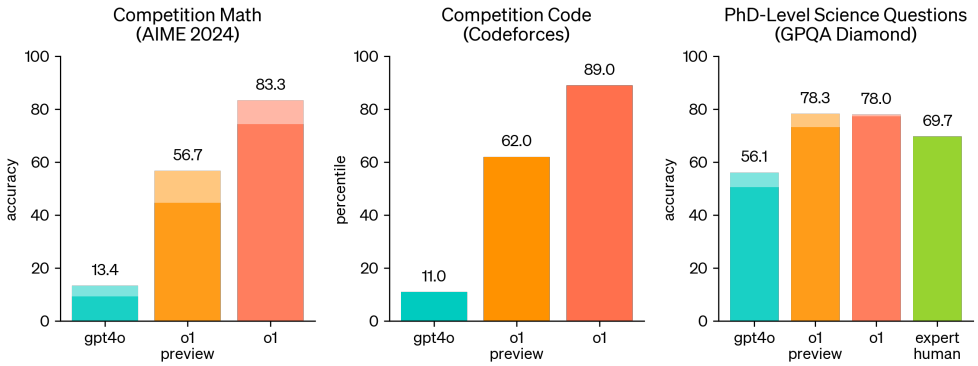


Figure 1: Impact of RL on LLM reasoning performance across math, coding, and scientific benchmarks.

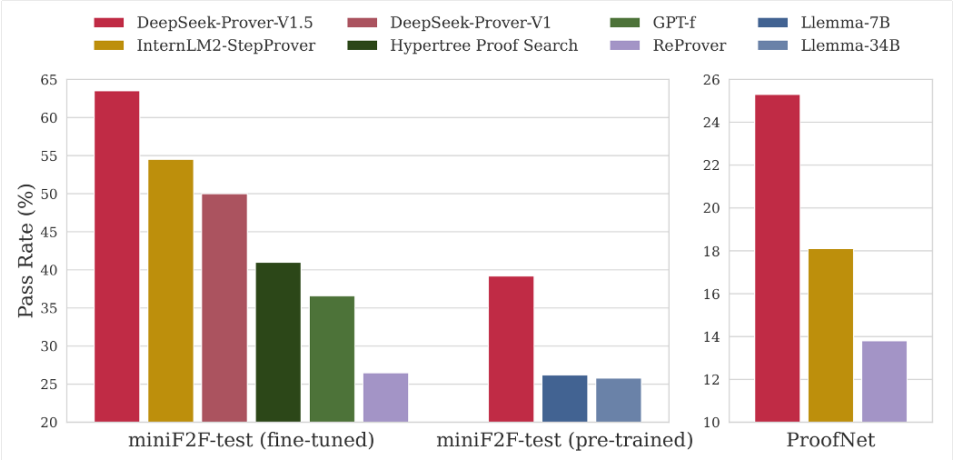


Figure 2: Performance of RL-enhanced theorem-proving models across miniF2F and ProofNet benchmarks (placeholder).

computing an estimator of the policy gradient and applying it within a Stochastic Gradient Ascent algorithm.

According to the Policy Gradient Theorem, we typically update the policy by optimizing the following objective function $L^{PG}(\theta)$:

$$L^{PG}(\theta) = \hat{E}_t \left[\log \pi_\theta(a_t|s_t) \hat{A}_t \right]$$

Here, $\pi_\theta(a_t|s_t)$ denotes the probability of taking action a_t given state s_t. To maximize this objective function, we must compute its gradient with respect to the parameter θ. Differentiating the objective function yields the gradient estimator $\hat{g}$:

$$\hat{g} = \hat{E}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t \right]$$

Where:

$$\hat{A}_t = Q_{\pi_\theta}(s_t, a_t) - V_{\pi_\theta}(s_t)$$

$V_{\pi_\theta}(s_t)$ serves as a Baseline. The primary motivation for introducing a baseline is twofold:

- Assessing Relative Advantage: It shifts the focus from the absolute return of an action to measuring the relative quality of a specific action a_t compared to the average performance in that state.
- Variance Reduction: As illustrated in the Fig. 3, original Q values may all be large positive numbers, leading to volatile gradient updates in a single direction. In contrast, the advantage

values obtained after baseline subtraction are distributed around zero with a clear distinction between positive and negative values. This process significantly reduces the variance of the estimator while maintaining the expected value of the gradient, thereby enhancing optimization stability.

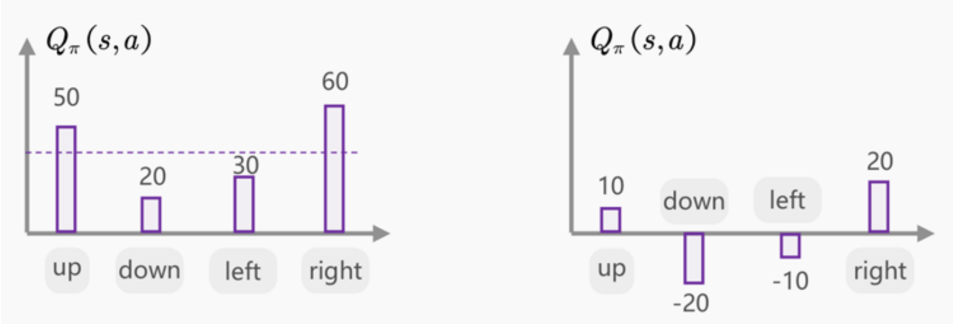


Figure 3: Illustration of the Advantage Function for Variance Reduction.

Standard PG is highly sensitive to the step size of parameter updates. If the step size is too large during a single update, the policy distribution π_θ may undergo drastic changes, driving the policy into a degenerate region of the parameter space. Once the policy deteriorates, the quality of subsequently collected data diminishes. This prevents the algorithm from recovering, ultimately precipitating a sudden collapse in performance during the training process.

3.2 Trust Region Policy Optimization (TRPO)

To address the challenge of determining appropriate update step sizes in standard Policy Gradient methods, Trust Region Policy Optimization proposes an optimization framework based on trust regions. The core idea is to shift away from directly controlling the step size of the parameter θ; instead, it constrains the difference between the policy distributions before and after the update, thereby ensuring a monotonic improvement in policy performance.

In the process of policy optimization, we typically aim to use data sampled from the old policy $\pi_{\theta_{old}}$ to estimate the expected return of the new policy π_θ. To facilitate this off-policy (or approximately off-policy) evaluation, TRPO introduces Importance Sampling techniques.

Importance Sampling allows us to compute expectations with respect to a distribution $p(x)$ while sampling from a different distribution $q(x)$ by introducing a probability density ratio. The mathematical identity is shown below:

$$E_{x\sim p(x)}[f(x)] = \int q(x)\frac{p(x)}{q(x)}f(x)dx = E_{x\sim q(x)}\left[\frac{p(x)}{q(x)}f(x)\right]$$

Based on this principle, TRPO reconstructs the optimization goal into a Surrogate Objective. This function no longer directly calculates expectations over trajectories under the new policy. Instead, it utilizes samples generated by the old policy $\pi_{\theta_{old}}$ and applies a weighted correction to the advantage function $\hat{A}_t$ using the ratio $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$. This mechanism enables the algorithm to effectively infer the performance of the new policy using historical data, maximizing the following expectation:

$$\hat{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right]$$

Although Importance Sampling addresses the data utilization issue, the variance of the estimator increases drastically when the difference between the new and old policies becomes too large, potentially causing the update to fail. To mitigate this, TRPO imposes a hard constraint while maximizing the surrogate objective: limiting the Kullback-Leibler (KL) Divergence between the new and old policies. This defines a Trust Region, ensuring that the magnitude of policy updates remains within a safe range.

The standard optimization problem in TRPO is formalized as a constrained non-linear programming problem:

$$\max_{\theta} \hat{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] \quad \text{subject to} \quad \hat{E}_t \left[KL\left(\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\right) \right] \leq \delta$$

Where δ represents the maximum allowable divergence threshold. This form of constraint effectively avoids performance collapse caused by excessively large step sizes. Furthermore, this problem can be converted into an unconstrained form by introducing a penalty term β via the method of Lagrange Multipliers, rewriting the optimization objective as:

$$\max_{\theta} \hat{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t - \beta KL\left(\pi_{\theta_{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)\right) \right]$$

.

3.3 Proximal Policy Optimization (PPO)

Although TRPO effectively guarantees monotonic policy updates through trust region constraints, its solution process involves second-order optimization, which is computationally expensive and difficult to implement. To reduce computational costs while maintaining algorithm stability, Proximal Policy Optimization was proposed. Like TRPO, PPO is based on Importance Sampling, but it foregoes complex KL divergence constraints in favor of a clipping mechanism—a first-order approximation—to limit the difference between the new and old policies.

The core innovation of PPO lies in the design of its objective function $L^{CLIP}(\theta)$. This function aims to maximize policy returns while penalizing updates that cause drastic changes in the policy distribution. Defining the probability ratio as $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, the optimization objective of PPO is:

$$L^{CLIP}(\theta) = \hat{E}_t \left[\min\left(r_t(\theta)\hat{A}_t^{GAE}, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^{GAE} \right) \right]$$

In the above equation, ϵ is a hyperparameter (typically set to 0.1 or 0.2) that defines the allowable fluctuation range. The clip function forces the ratio $r_t(\theta)$ to stay within the interval $[1-\epsilon, 1+\epsilon]$. By introducing the $\min$ operator, the algorithm effectively selects the lower value between the unclipped original objective and the clipped restricted objective. This design ensures that high rewards are obtained only when policy updates are within a safe range; once the update magnitude is excessive (i.e., the ratio r deviates too far from 1), the objective function value is truncated, thereby eliminating the incentive for excessively large step sizes.The specific behavior of the clipping mechanism depends on the sign of the advantage function $\hat{A}_t$, as illustrated in the figure below. When $\hat{A}_t > 0$, indicating the current action is better than average, we need to increase the probability of that action, but the growth of the ratio r_t is capped at $1+\epsilon$ to prevent policy drift caused by overconfidence. Conversely, when $\hat{A}_t < 0$, we need to decrease the probability, but the reduction of r_t is limited to $1-\epsilon$ to prevent the probability from dropping instantly to zero, which would destroy exploration. This mechanism simply yet effectively simulates the trust region constraints of TRPO.

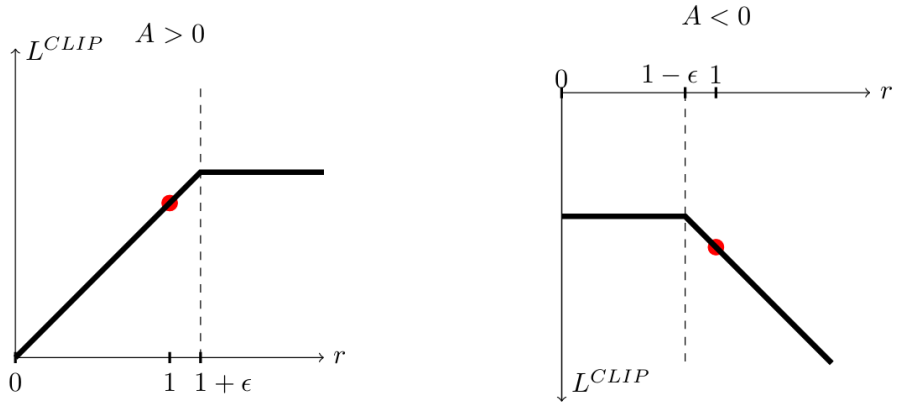


Figure 4: PPO Objective Function Curve.

To further balance variance and bias, PPO is typically used in conjunction with Generalized Advantage Estimation (GAE). GAE is analogous to Multi-Step Temporal Difference learning; it estimates advantage values using a weighted accumulation of Temporal Difference errors δ_t:

$$A_t^{GAE} = (1-\lambda)(A_t^{(1)} + \lambda A_t^{(2)} + \lambda^2 A_t^{(3)} + \dots) = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

Where δ_t represents the TD error. By adjusting the parameter λ, GAE can flexibly trade off between high bias and high variance, much like the trade off between Monte Carlo method and Temporal Difference, thereby providing the PPO algorithm with a more accurate and stable gradient estimation signal.

3.4 Group Relative Policy Optimization (GRPO)

To further reduce the consumption of computational resources and eliminate the dependency on a separate independent value function approximator, Group Relative Policy Optimization was proposed. It leverages the statistical properties of multiple output samples generated from the same input to directly estimate advantage values.

The core mechanism of GRPO lies in its unique method for calculating the advantage function. For each input prompt s, the policy network $\pi_{\theta_{old}}$ samples a group of G distinct output trajectories $\{o_i\}_{i=1}^{G}$. Subsequently, the reward model calculates the reward value r_i for each output, then uses the mean and standard deviation of this group of rewards to standardize the rewards. Specifically, the advantage value $\hat{A}_{i,t}$ for the i-th output at time t is defined as the standardized reward $\tilde{r}_i$:

$$\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(r)}{\text{std}(r)}$$

Where $\text{mean}(r)$ and $\text{std}(r)$ represent the average and standard deviation of the G output rewards within that group, respectively.

The overview of the difference between PO and GRPO is illustrated in Figure 5, While PPO requires maintaining and updating four separate models (Policy, Reference, Reward, and Value), GRPO removes the need for the Value Model, thereby reducing memory usage and computational overhead during training.

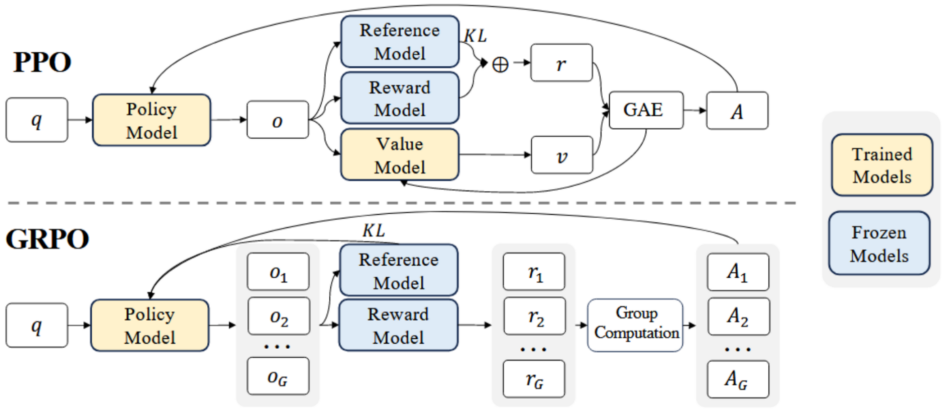


Figure 5: Architecture comparison between PPO and GRPO.

The overall optimization objective of GRPO, $L^{GRPO}(\theta)$, inherits the clipping concept from PPO but performs averaging at the group level. For each input s, the algorithm maximizes the following objective:

$$L^{GRPO}(\theta) = \hat{E}_t \left[\frac{1}{G} \sum_{i=1}^{G} \left(\min \left(\frac{\pi_\theta(a_i|s)}{\pi_{\theta_{old}}(a_i|s)} \hat{A}_{i,t}, \text{clip} \left(\frac{\pi_\theta(a_i|s)}{\pi_{\theta_{old}}(a_i|s)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_{i,t} \right) - \beta KL[\pi_{\theta_{ref}}(\cdot|s_i) || \pi_\theta(\cdot|s_i)] \right) \right]$$

To prevent the model from deviating excessively from the original (or reference) model π_{ref} during optimization, a KL divergence term is typically integrated into the reward signal r_t:

$$r_t = r_\phi(s, a_{\leq t}) - \beta \log \frac{\pi_\theta(a_t|s, a_{<t})}{\pi_{ref}(a_t|s, a_{<t})}$$

4 Introduction to Galaxian

PPO has been shown to have great performance over many Atari tasks, while the performance of GRPO is unclear at this stage. Since GRPO is initially designed for tuning Large Language Model (LLMs), we may select some of the Atari games that are more similar to the LLM landscape, e.g., reward density, reward variance in one group. Implementing GRPO on these games is expected to be non-trivial. On the other hand, the potential limitation of GRPO can also be explored on Atari games which are very different from LLM landscape.

4.1 Game Rules

Galaxian is a fixed-shooter arcade game where the player controls a spaceship at the bottom of the screen.

- **Objective:** Destroy alien formations and survive.
- **Lives:** Start with 3 lives; extra lives awarded at score thresholds.
- **Enemy Behavior:** Aliens remain in formation but periodically dive toward the player.
- **Scoring:** Higher-tier enemies yield more points; destroying diving enemies gives bonus.

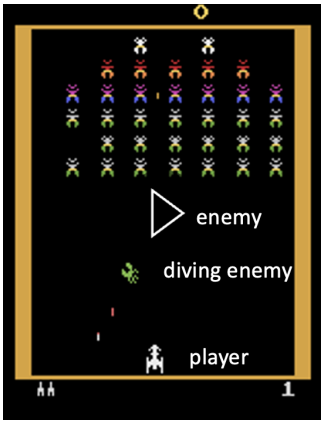


Figure 6: Galaxian gameplay illustration.

4.2 Implementation Details

- **Environment:** ALE "Galaxian-v5"
- **Observation:** $210 \times 160 \times 3$ RGB frames
- **Action space:** 6 discrete actions (NOOP, FIRE, LEFT, RIGHT, LEFT+FIRE, RIGHT+FIRE)
- Each `env.step()` advances the game by exactly one frame.

5 Results

5.1 Training

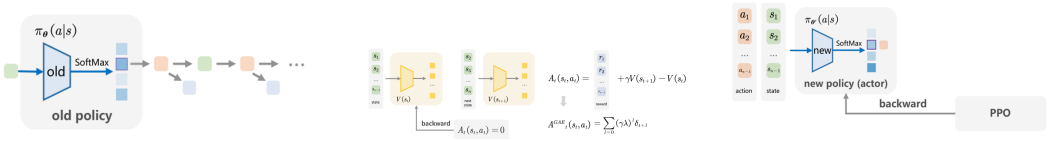
In this experiment, we employ the Proximal Policy Optimization (PPO) algorithm. The training process follows a closed-loop structure consisting of sampling, evaluation, and optimization, as illustrated in Figure 7.

The algorithm begins with the data sampling stage(Figure 7a). In this stage, the current "Old Policy" ($\pi_{\theta_{old}}$) interacts with the environment to generate trajectory sequences. The system records tuples containing the state s_t, executed action a_t, environmental reward r_t, and the corresponding probability distribution of the action. These real-time interaction data form the experience buffer, serving as the foundation for subsequent policy optimization.

Subsequently, the agent calculate the GAE function (Figure 7b). To assess the quality of current actions, a value network (Critic) is introduced to estimate the state value $V(s)$. Based on the critic's output, the Generalized Advantage Estimation (GAE) is employed to compute the advantage function A_t^{GAE}. As shown in the equations, GAE balances bias and variance by utilizing a discount factor γ and a smoothing parameter λ, thereby providing lower-variance gradient estimates for policy updates.

Finally, the agent preforms policy update (Figure 7c). Using the same trajectory The model iteratively updates the policy network (Actor) using the PPO-Clip objective function, and update value network to move advantage function towards zero. The same trajectory is used to perform several epoches of optimization, then goes to the data sampling stage iteratively.

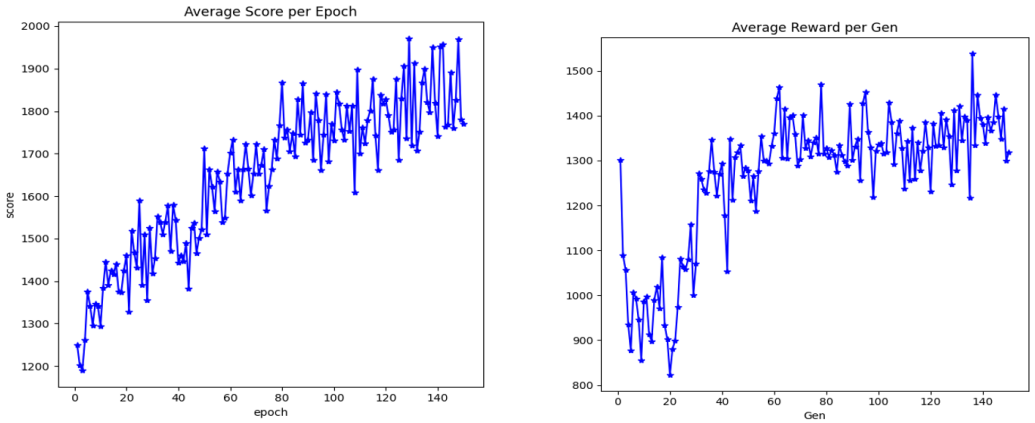
GRPO training follows the similar process, except that it generate a group of trajectory, and use the group average to calculate the GAE function.



(a) Data Sampling (b) Advantage Estimation (c) Policy Update

Figure 7: **The PPO Training Pipeline.**

To ensure a fair comparison, both algorithms were trained for 150 iterations under identical hyper-parameter settings.The baseline configuration included a learning rate of 1×10^{-4} and a discount factor γ of 0.99. For data collection, 16 parallel actors were employed, with each actor processing 1024 frames per iteration. For the optimization of PPO, parameter θ updates were performed using 4 epochs and a batch size of 32. GRPO retained these baseline hyperparameters, with its specific group size set to 16.



(a) PPO Training Curve (b) GRPO Training Curve

Figure 8: Comparison of training performance between PPO and GRPO.

5.2 Performance

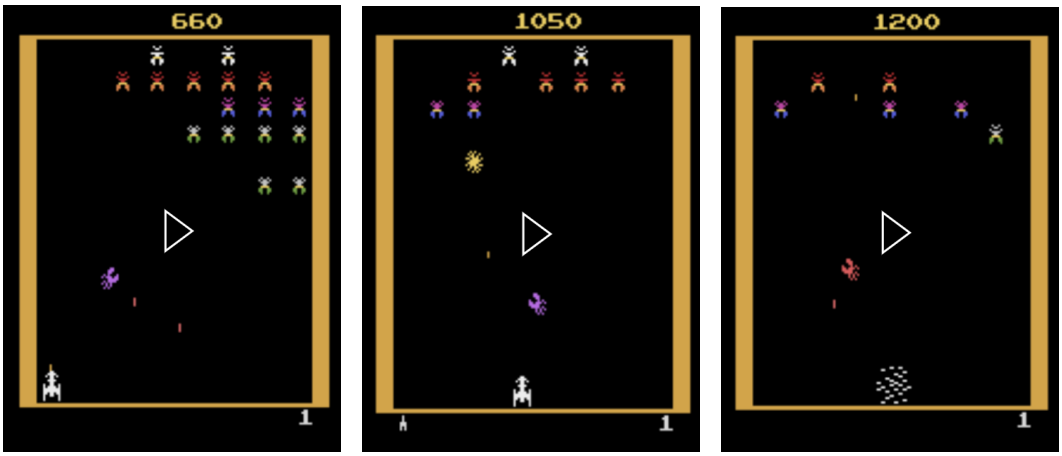
The training curves reveal distinct learning trajectories and performance characteristics for the two algorithms.

PPO demonstrated superior training stability and convergence speed. As illustrated, the average score per epoch for PPO exhibits a clear and continuous upward trend. Starting from an initial score of approximately 1200, the model's performance improved steadily with the number of iterations, showing no significant oscillation. By the final (150th) iteration, the average score for PPO exceeded 1800, maintaining low variance in the high-score region. This demonstrates the robustness of the policy update mechanism within the current environment.

In contrast, GRPO exhibited considerable instability over the same period. Although its initial average Reward was approximately 1300—slightly higher than PPO's starting level—GRPO experienced a significant performance collapse during the early training phase (approximately the first 20 iterations), with the average reward briefly dropping below 900. While the model successfully escaped the local minimum and began to recover, exhibiting a "dip-then-rise" trend, the subsequent rate of improvement was gradual. By the end of training, GRPO's final convergence score remained around 1400, significantly lower than the 1800+ achieved by PPO. This suggests that under the current configuration (Group Size = 16), GRPO's sample efficiency and policy update stability are inferior to PPO for this task, with the training process characterized by greater variance fluctuations.

5.3 Behavioral Analysis

To validate the performance disparities observed in the training curves, we further analyzed the models' actual gameplay behavior on the validation set:



(a) Random Policy (b) GRPO Agent (c) PPO Agent

Figure 9: Visualization of actual gameplay behavior across three policies.

Random Policy (Baseline, Score: 660): Serving as a baseline control, the random policy exhibited complete disorder. The agent lacked effective environmental perception, failing to track targets or evade damage. This resulted in rapid game termination and the lowest score of 660.

GRPO (Intermediate, Score: 1050): The GRPO policy demonstrated evidence of learning, outperforming the random policy and indicating that the agent had acquired basic survival and attack logic. However, consistent with the oscillations and lower convergence values observed in the training curve, GRPO's in-game performance appeared relatively conservative and inefficient. Compared to PPO, it failed to maximize the score within the same decision window and appeared more prone to suboptimal states, resulting in a significantly lower accumulated reward.

PPO (Superior, Score: 1200): The PPO policy achieved the best performance with a score of 1200. This aligns closely with the stable and continuously rising learning curve observed during training. Visually, the PPO agent demonstrated higher scoring efficiency than GRPO, effectively neutralizing enemy units to rapidly accumulate points. These results confirm that, under the current parameter settings, PPO possesses superior policy robustness and sample efficiency.

6 Discussion

Our results highlight a fundamental limitation of GRPO when applied to traditional reinforcement learning tasks. Unlike in large language models, where both the actor and the critic originate from a powerful pretrained system with strong semantic understanding, Atari agents begin with a randomly initialized CNN that has no meaningful feature representations. Without a well-formed reference model, the groupwise comparisons used by GRPO provide little useful learning signal. In early training, all sampled actions and trajectories appear equally uninformative, causing GRPO to simply select among "junk" outputs.

In contrast, PPO benefits from a value function that gradually shapes the feature extractor, offering supervised guidance that helps the network learn to distinguish relevant visual patterns, such as identifying enemies or player states. This explains why PPO improves steadily while GRPO stagnates.

Overall, GRPO's effectiveness in LLM reasoning tasks relies heavily on strong pretrained priors, where candidate outputs are coherent enough for relative ranking to be meaningful. In environments without such priors—like vision-based Atari games—GRPO lacks the structured signal needed to guide learning, limiting its applicability outside pretrained LLM settings.

7 Conclusion

This study evaluated the applicability of Group Relative Policy Optimization (GRPO) in traditional reinforcement learning tasks by comparing it with Proximal Policy Optimization (PPO) within the Atari Galaxian environment. Experimental results demonstrate that while GRPO excels in Large Language Model (LLM) fine-tuning, its convergence speed and final policy performance are significantly inferior to PPO in Atari tasks initiated from random parameters.

Based on our experimental data and theoretical analysis, we attribute the limitations of GRPO in traditional RL tasks to the absence of a strong reference model and the specific mechanisms of feature extraction. In LLM training, RLHF is typically applied to pretrained models that have undergone Supervised Fine-Tuning (SFT) and already possess robust representation capabilities. In contrast, Atari agents utilize Convolutional Neural Networks (CNNs) initialized with random weights. PPO facilitates representation learning by training a value function, where the regression loss serves as a supervision-like signal for the feature extraction layers.

Conversely, GRPO relies on the relative advantage of samples within a group for updates. In the early stages of training where the model lacks prior knowledge, the generated sample groups are predominantly low-quality, hindering the algorithm's ability to identify effective directions for policy improvement. In conclusion, GRPO is inherently dependent on the initial capabilities of the base model. While it is suitable for the alignment and optimization of pre-existing capabilities, its effectiveness is limited in traditional end-to-end reinforcement learning tasks that require constructing both feature extraction and decision-making policies from scratch.

References

[1] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[3] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[4] Huajian Xin, ZZ Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, et al. Deepseek-prover-v1. 5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *arXiv preprint arXiv:2408.08152*, 2024.