Comparative Analysis of PPO, DQN and PPG in the Procgen benchmark

Sushant Karki
Science Academy
University of Maryland College Park
skarki@umd.edu

Chandramani
Science Academy
University of Maryland College Park
cm05@umd.edu

***Abstract*—This report delves into how well different learning methods perform in the Starpilot game within the Procgen Benchmark, focusing on Proximal Policy Optimization (PPO), Deep Q-Network (DQN), and Phasic Policy Gradient (PPG). The goal is to understand how these algorithms generalize in this gaming scenario. Interestingly, our study discovers that PPG stands out as the most effective among the tested algorithms for this particular game. This insight contributes to the ongoing exploration of reinforcement learning and its applications in diverse gaming environments. All three algorithms were trained for 5,000,000 steps in the same number of environments. At the end of training, for PPO, the average episodic length was observed at 145.0 with an average episodic return of 8.0. In contrast, PPG demonstrated a significantly higher performance with an average episodic length of 577.0 and an average episodic return of 51.0. Although specific statistics for DQN are not currently available, preliminary observations suggest its performance does not match that of PPG. This comparative analysis within the Procgen environment highlights the superior sample efficacy of PPG in environments demanding high diversity and visual recognition. The results provide valuable insights into the sample efficiency and efficacy of different reinforcement learning algorithms in dynamic and complex settings. The study contributes to the broader understanding of algorithmic performance in reinforcement learning, offering practical guidance for future algorithm selection and optimization in similar environments.**

***Index Terms*—Reinforcement Learning, DQN, PPG**

I. Introduction

The field of reinforcement learning, with its continuous evolution and expanding applications, presents a myriad of challenges and opportunities for exploration. A key area of interest is the development of algorithms capable of efficient learning and generalization across diverse environments. The Procgen Benchmark, comprising 16 distinct environments, offers an ideal platform for such explorations, particularly in evaluating the adaptability and effectiveness of various reinforcement learning algorithms. This study focuses on the Starpilot game within the Procgen Benchmark, a game that presents a variety of different challenges that require an algorithm to generalize well in order to perform well. Starpilot requires a combination of strategic learning, visual recognition, and motor control, making it a challenging environment for testing reinforcement learning algorithms. Our study is an initial attempt in comparing the performance of three algorithms in this setting: Proximal Policy Optimization (PPO), Deep Q-Network (DQN), and Phasic Policy Gradient (PPG). These algorithms were selected for their prominence and ease of use in similar environments. The primary aim of this study is to provide an introductory analysis of how these algorithms perform in a dynamic and visually complex environment like Starpilot. We seek to understand their basic learning behaviors and generalization capabilities in this context. This work is not intended as a comprehensive or exhaustive exploration but rather as a preliminary step in understanding the challenges and possibilities in applying these algorithms in complex environments. Given the scope and exploratory nature of our study, the findings should be considered as initial observations that contribute to a broader conversation about algorithm effectiveness in reinforcement learning. This research is a modest attempt to add to the existing knowledge base, providing insights that may inform more extensive future studies. Our work highlights the importance of continued research in this field, particularly in developing and testing algorithms that can efficiently learn and adapt to new, diverse scenarios.

II. Algorithms and Environment

A. Algorithms

1) Deep Q-Network (DQN): The Deep Q-Network (DQN) algorithm, unveiled by Mnih et al. in 2013 [1], stands out as a major breakthrough in the realm of deep reinforcement learning. It marries the principles of Q-learning, a well-regarded technique in reinforcement learning, with the power of deep neural networks. This marriage empowers agents to grasp effective strategies even when faced with complex, high-dimensional inputs, such as raw pixel data. DQN introduces clever concepts like experience replay and fixed Q-targets, playing a crucial role in steadying the learning process. Its success is evident in tackling a variety of Atari 2600 games, showcasing its prowess in environments demanding visual understanding and intricate strategic thinking. This pioneering approach has significantly influenced how machines learn and adapt in dynamic, visually rich scenarios.

2) Proximal Policy Optimization (PPO): Proximal Policy Optimization (PPO), created by Schulman and team [2], stands out as a valuable method in the realm of reinforcement learning. Its main goal is to tackle challenges related to sample efficiency and training stability, particularly in intricate environments. PPO takes a unique approach by optimizing a surrogate objective function. This optimization is done with

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Fig. 1. DQN Algorithm

careful consideration to prevent the new policy from straying too far from the old policy. Striking this balance between exploring new possibilities and exploiting existing knowledge has contributed to PPO's widespread adoption in various fields, such as robotic control and game playing. Its versatility and ability to navigate complex scenarios make it a popular choice for those seeking effective solutions in reinforcement learning.

Algorithm 5 PPO with Clipped Objective

Input: initial policy parameters θ_0, clipping threshold ϵ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}^{CLIP}(\theta)$$

  by taking K steps of minibatch SGD (via Adam), where

$$\mathcal{L}_{\theta_k}^{CLIP}(\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_k} \left[\sum_{t=0}^{T} \left[\min(r_t(\theta)\hat{A}_t^{\pi_k}, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t^{\pi_k}) \right] \right]$$

end for

Fig. 2. PPO Algorithm

3) Phasic Policy Gradient (PPG): Phasic Policy Gradient, brought into play by Cobbe and team in 2020 [3], is a cool algorithm meant to make learning better and steadier when it comes to training policies in reinforcement learning. What's neat about PPG is that it breaks down the updating of policies and value functions into different phases. This split-up way of doing things makes learning more steady and efficient, which is super helpful, especially when you're dealing with environments that have lots of complex stuff going on. So far, it's shown good results in different situations, hinting that it could be a pretty solid way to learn things. And just so you know, we're still working on this, so stay tuned for more updates!

B. Environment

Procgen : The Procgen Benchmark, developed by Cobbe et al., is a collection of procedurally generated environments designed to test and compare the robustness and generalization capabilities of reinforcement learning algorithms. Among these environments, Starpilot stands out as a particularly challenging and illustrative testbed. [4]

Algorithm 1 PPG

for phase $= 1, 2, \ldots$ **do**
  Initialize empty buffer B
  for iteration $= 1, 2, \ldots, N_\pi$ **do** ▷ Policy Phase
    Perform rollouts under current policy π
    Compute value function target $\hat{V}_t^{\text{targ}}$ for each state s_t
    for epoch $= 1, 2, \ldots, E_\pi$ **do** ▷ Policy Epochs
      Optimize $L^{clip} + \beta_S S[\pi]$ wrt θ_π
    for epoch $= 1, 2, \ldots, E_V$ **do** ▷ Value Epochs
      Optimize L^{value} wrt θ_V
    Add all $(s_t, \hat{V}_t^{\text{targ}})$ to B
  Compute and store current policy $\pi_{\theta_{old}}(\cdot|s_t)$ for all states s_t in B
  for epoch $= 1, 2, \ldots, E_{aux}$ **do** ▷ Auxiliary Phase
    Optimize L^{joint} wrt θ_π, on all data in B
    Optimize L^{value} wrt θ_V, on all data in B

Fig. 3. PPG Algorithm

StarPilot in Procgen Benchmark Starpilot is a fast-paced, space-themed shooting game within the Procgen Benchmark suite. In this game, the player controls a spaceship navigating through a dynamically changing environment filled with obstacles and adversaries. The primary objectives are to avoid collisions with obstacles, shoot down enemies, and survive as long as possible while maximizing the score.

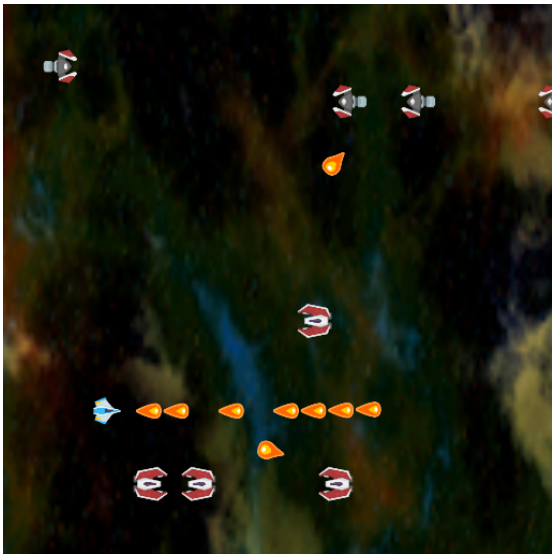

Fig. 4. Starpilot in Procgen Benchmark

1) Key Challenges of Starpilot:

- **Dynamic Environment:** The procedural generation in Starpilot ensures that each playthrough is unique, with different configurations of obstacles and enemies. This unpredictability requires the learning algorithm to be highly adaptable and capable of generalizing from past experiences to unseen scenarios.
- **Visual Complexity:** The game features rich and varied visual elements. The agent needs to process and interpret complex visual inputs to make real-time decisions, challenging the algorithm's capacity for visual recognition and decision-making under pressure.

- **Strategic Decision-Making:** Success in Starpilot involves not just reactive skills but also strategic planning. The agent must learn to balance between aggressive actions (like shooting enemies) and defensive maneuvers (like avoiding obstacles), demanding a nuanced understanding of the game's mechanics and strategy.
- **Continuous Control and Precision:** The game requires continuous control over the spaceship, with a need for precise movements to navigate through tight spaces and evade enemy fire. This aspect tests the algorithm's ability in fine motor control and continuous action spaces.

In the context of robotics, Starpilot is a comprehensive test of an algorithm's capabilities in dealing with complex, dynamic, and visually rich environments. It pushes the boundaries of what these algorithms can achieve in terms of learning efficiency, adaptability, and generalization. By evaluating algorithms in Starpilot, researchers can gain valuable insights into their strengths and weaknesses, informing future developments in the field.

III. Methodology

The methodology of this study is structured to evaluate and compare the performance of three reinforcement learning algorithms — Proximal Policy Optimization (PPO), Deep Q-Network (DQN), and Phasic Policy Gradient (PPG) — within the Starpilot game of the Procgen Benchmark. The focus is on assessing these algorithms in terms of learning efficiency and adaptability in a complex, procedurally generated environment.

A. Environment Setup

- **Procgen Benchmark - Starpilot:** The Starpilot game, part of the Procgen Benchmark, was chosen for its dynamic challenges and requirement for strategic decision-making.
- **Algorithm Implementation:** Adaptations of the CleanRL [5] implementations of PPO, DQN, and PPG algorithms were reimplemented to suit the unique requirements and the vectorized environments of the Starpilot game.

B. Training Process

- **Agent Training** 3 Agents were trained using each algorithm for a total of 5,000,000 steps, and similar hyperparameters ensuring a comprehensive assessment of learning and adaptability.

  Hyperparamters for Each Algorithms
  - **Training steps** : 5M
  - **Parallel envs (Procgen specific)** : 64
  - **Learning Rate** : PPO, PPG : 5e-4 , DQN: 1e-4
  - **Gamma (Discount Factor):** PPO, PPG: 0.999, DQN: 0.99
  - **Observation space:** RGB values of each pixel on the screen of the star pilot game.
  - **Action Space :** Our action space is discrete, meaning we have a set of distinct actions to choose from. The environment provides 15 actions, but in practice, we use only 11 of them because there are 5 actions that essentially do the same thing. This simplifies our decision-making process and streamlines the actions we take in the given environment.

C. Evaluation Metrices

The study used the average episodic length and episodic return as the main evaluation metrics. These metrics were chosen to assess the agents' generalization ability and sample efficiency within the game environment.

D. Hypothesis

- **Deep Q-Network (DQN):** While there may be a necessity for an extended training duration, we anticipate that DQN will ultimately converge, demonstrating its capability to learn effective policies.
- **Proximal Policy Optimization (PPO):** Our expectation is that PPO will exhibit a consistent and stable improvement in evaluation metrics, showcasing its robustness in handling complex learning environments.
- **Phasic Policy Gradient (PPG):** We hypothesize that PPG, while achieving performance on par with PPO, will do so within a notably reduced number of training timesteps. This assumption is grounded in PPG's design, aiming for enhanced efficiency and quicker convergence.

E. Limitation and Consideration

- **Training Duration:** The training of each algorithm was limited to 5,000,000 steps, which could impact the depth of learning and adaptability. In comparison, identical studies on the Procgen environment in the past by other people showed training over 200M+ steps and achieving some kind of convergence compared to which our training duration was very small.
- **Scope of Study:** As the study is confined to the Starpilot game, the findings may not be entirely applicable to other games within the Procgen Benchmark or different reinforcement learning contexts.

IV. Results

Our experimental results reveal distinctive performance characteristics among the evaluated reinforcement learning algorithms. The Deep Q-Network (DQN) exhibits an average episodic length of 106.0 steps and achieves an average episodic return of 6.0, showcasing its ability to complete episodes efficiently but with a modest overall return. PPG, on the other hand, demonstrates a substantially longer average episodic length at 577.0 steps, coupled with a higher average episodic return of 51.0. This suggests that PPG, while taking more steps on average, achieves a more favorable overall return compared to DQN. Proximal Policy Optimization (PPO) strikes a balance between the two, with an average episodic length of 145.0 steps and an average episodic return of 8.0. The uniformity in training steps, set at

Algorithm	Avg. Episodic Length	Avg. Episodic Return
DQN	106.0	6.0
PPG	577.0	51.0
PPO	145.0	8.0

TABLE I
EXPERIMENTAL RESULTS FOR REINFORCEMENT LEARNING ALGORITHMS

5,000,000 for all algorithms, provides a standardized basis for comparison, allowing us to discern their relative efficiency and effectiveness in achieving the specified learning objectives. These results offer valuable insights into the trade-offs and strengths of each algorithm in navigating the learning environment.:

A. Average Episodic Length

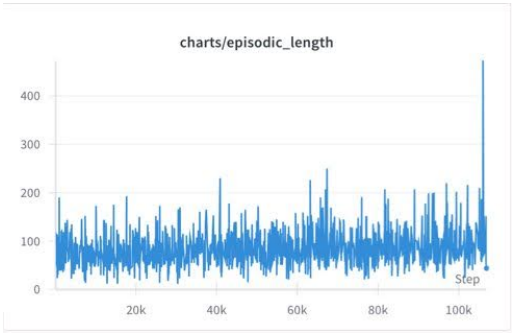

Fig. 5. Progression of Avg. Episodic Length of DQN

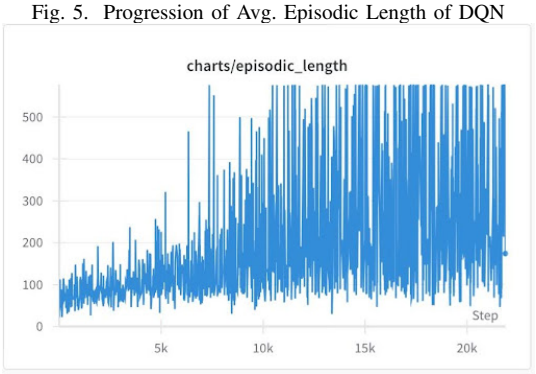

Fig. 6. Progression of Avg. Episodic Length of PPO

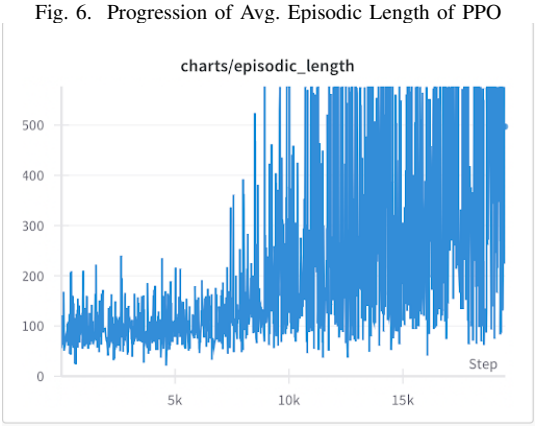

Fig. 7. Progression of Avg. Episodic Length of PPG

B. Average Episodic Return

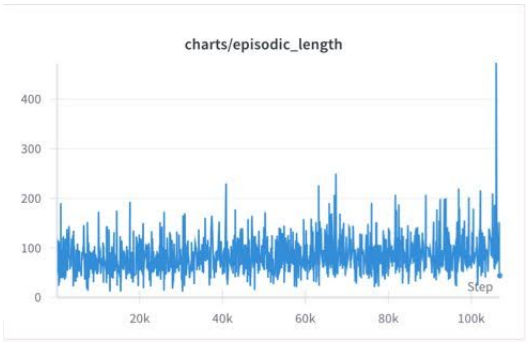

Fig. 8. Progression of Avg. Episodic Return of DQN

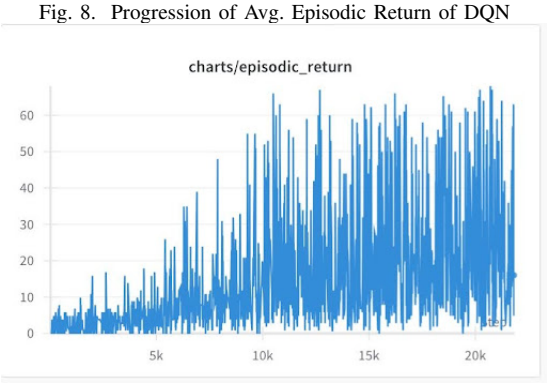

Fig. 9. Progression of Avg. Episodic Return of PPO

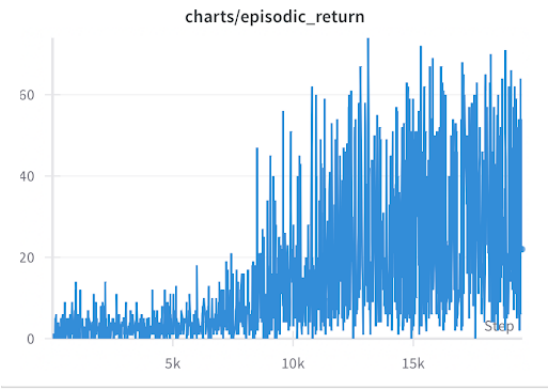

Fig. 10. Progression of Avg. Episodic Return of PPG

V. CONCLUSION

In conclusion, our study delved into the comparison of three reinforcement learning algorithms—Proximal Policy Optimization (PPO), Deep Q-Network (DQN), and Phasic Policy Gradient (PPG)—using the Starpilot game within the Procgen Benchmark. Notably, the results underscored PPG's exceptional performance in terms of sample efficiency and generalization, surpassing the other algorithms, namely PPO and DQN. The success of PPG in navigating the complexities of this particular environment showcases its adaptability, presenting a contrast to the comparatively weaker outcomes observed with PPO and DQN.

It's essential to acknowledge the study's limitations, primarily its focus on a single game and a relatively brief training duration. This limitation may not fully capture the algorithms' long-term learning potential and broader applicability across diverse scenarios. Despite these constraints, our findings provide valuable insights into the strengths of different reinforcement learning approaches within complex settings. They underscore the critical aspect of selecting algorithms based on specific task requirements. This study contributes to the ongoing development of the field, enriching our understanding of reinforcement learning in dynamic environments.

VI. Future Work

Moving forward, our exploration into reinforcement learning algorithms prompts considerations for future research avenues. One intriguing direction involves expanding the evaluation across multiple games within the Procgen Benchmark, encompassing diverse challenges and environmental characteristics. This broader exploration could offer a more comprehensive understanding of how PPG's exceptional performance extends to various gaming scenarios. Additionally, extending the training duration for all algorithms beyond the current 5,000,000 steps may unveil nuanced patterns in long-term learning and generalization capabilities.

Furthermore, investigating the transferability of PPG's success to real-world applications remains an exciting prospect. The insights gained from its superior sample efficiency and adaptability in a gaming environment could potentially be leveraged for real-world tasks with high diversity and visual recognition requirements. This transition from simulated environments to real-world scenarios is a crucial step in validating the practical utility of reinforcement learning algorithms.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

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

[3] Karl Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient, 2020.

[4] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning, 2020.

[5] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.