Training Q-learning and PPO Agents in the MsPacman-v5 Environment

Wei Syuan Liao
Department of Computer Science
National Tsing Hua University (NTHU)
floram511jb@gmail.com

Abstract

This report details the process and outcomes of training Q-learning and PPO agents in the MsPacman-v5 environment. The objective was to surpass a baseline performance of a random agent with an average reward of 221. While the Q-learning agent faced challenges due to the high-dimensional state space and complex game dynamics, achieving an average reward of 326, the PPO agent demonstrated significant success by achieving an average reward of 967. These findings highlight the robustness of policy-based methods like PPO in complex environments and underscore the need for advanced techniques to enhance the performance of value-based methods such as Q-learning.

1 Introduction

Deep Reinforcement Learning (DRL) has emerged as a powerful tool for training agents to perform complex tasks by learning from interactions with their environment. This capability has been leveraged in various domains, including robotics, autonomous systems, and game playing, where agents must navigate dynamic and high-dimensional spaces to achieve specific goals. Among the numerous environments available for testing DRL algorithms, the MsPacman-v5 environment from OpenAI's gym library stands out due to its complexity and richness in game states and interactions.

The primary objective of this project is to train agents using Q-learning and Proximal Policy Optimization (PPO) algorithms in the MsPacman-v5 environment and compare their performance against a baseline random agent. Q-learning, a value-based method, and PPO, a policy-based method, represent two distinct approaches in reinforcement learning, each with its strengths and challenges. The random agent, serving as a baseline, helps in quantifying the effectiveness of these algorithms by providing a reference performance level.

In this study, we aim to explore how these algorithms cope with the challenges presented by the MsPacman-v5 environment. Specifically, we investigate the learning curves of both agents, analyze their performance metrics, and provide insights into the effectiveness of each method. The findings from this project can contribute to a better understanding of the suitability of different reinforcement learning approaches in handling complex environments, and guide future research in optimizing these algorithms for practical applications.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

2 Related Works

2.1 Q-learning

Q-learning is a value-based, model-free reinforcement learning algorithm. It learns the value of state-action pairs (Q-values) to make decisions. The update rule for Q-values is given by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{1}$$

where α is the learning rate, r is the reward received, γ is the discount factor, and s' is the next state. This formula iteratively updates the Q-values to reflect the expected future rewards of taking a given action in a given state. Q-learning has been widely applied in domains such as robotics, game playing, and autonomous systems, where it helps in making sequential decisions under uncertainty [2].

2.2 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a policy-based, model-free reinforcement learning algorithm that optimizes the policy directly using a clipped objective function to ensure stability. The objective function is defined as:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \tag{2}$$

where $r_t(\theta)$ is the probability ratio between the new and old policies, $\hat{A}_t$ is the advantage estimate, and ϵ is a hyperparameter that controls the clipping range. PPO has been effectively used in various tasks including robotics, game playing, and continuous control, providing a robust method for policy optimization that balances exploration and exploitation while maintaining stable learning updates [3].

2.3 MsPacman-v5 Environment

The MsPacman-v5 environment, provided by OpenAI's gym library, presents a complex and dynamic environment with various game states and interactions. It requires the agent to navigate through a maze, avoid ghosts, and collect pellets, which makes it an ideal testbed for evaluating deep reinforcement learning algorithms. The environment's complexity and stochastic nature challenge the agent's ability to learn effective strategies and adapt to changing game dynamics [1].

3 Methodology

3.1 Environment Setup

The MsPacman-v5 environment was configured with a state representation consisting of 4 stacked frames, each resized to 64x64 pixels, to capture the temporal dynamics of the game. The action space includes 9 possible actions that the agent can take, corresponding to different movement directions and actions within the game. This setup ensures that the agent has a comprehensive view of the environment's state and can make informed decisions based on recent observations.

3.2 Q-learning Implementation

The Q-learning agent was implemented with a state space size of 1,000,000 to manage the large number of possible game states through hashing. The action space consists of 9 possible actions, and the Q-table was initialized with zeros. The learning rate was set to 0.01 to control the rate of Q-value updates, and the discount factor was set to 0.99 to balance immediate and future rewards. The epsilon parameter, which controls the exploration-exploitation tradeoff, was initially set to 1.0 and decayed to 0.001 over time. The agent was trained for a total of 8,000 episodes, with each episode allowing a maximum of 200 steps. This setup aimed to balance learning stability and exploration efficiency.

3.3 PPO Implementation

The PPO agent was implemented using a neural network with a convolutional architecture designed to process the high-dimensional input frames from the MsPacman-v5 environment. The network

consists of three convolutional layers, followed by a fully connected layer. The first convolutional layer has 12 input channels and 32 output channels with an 8x8 kernel and stride of 4, the second layer has 32 input channels and 64 output channels with a 4x4 kernel and stride of 2, and the third layer has 64 input channels and 64 output channels with a 3x3 kernel and stride of 1. These layers are followed by a fully connected layer with 22528 input features and 512 output features, which then branches into the value head, alpha head, and beta head. The value head predicts the state value, while the alpha and beta heads, with 9 output features each, predict the parameters of the Beta distribution for the policy. The network parameters were optimized using the Adam optimizer with a learning rate of 0.001. The discount factor was set to 0.99, and the PPO-specific parameters included a clip parameter of 0.1, 10 PPO epochs per update, a buffer capacity of 2000 transitions, and a batch size of 128. This architecture and training setup aimed to provide a stable and efficient policy optimization process.

Layer Type	Input Channels	Output Channels	Kernel Size	Stride	Activation
Conv Layer 1	12	32	8x8	4	ReLU
Conv Layer 2	32	64	4x4	2	ReLU
Conv Layer 3	64	64	3x3	1	ReLU
Fully Connected Layer	22528	512	-	-	ReLU
Value Head	512	1	-	-	None
Alpha Head	512	9	-	-	Softplus
Beta Head	512	9	-	-	Softplus

Table 1: PPO Network Architecture

4 Experiments

4.1 Training Procedure

The Q-learning agent was trained for a total of 8,000 episodes, while the PPO agent was trained for 250 episodes. During training, the Q-learning agent used an ϵ-greedy policy to balance exploration and exploitation. The PPO agent utilized a Beta distribution to sample continuous actions, which were then discretized to match the discrete action space of the environment. The training process involved repeatedly resetting the environment, selecting actions based on the current policy, updating the policy based on observed rewards, and storing transitions for future updates.

4.2 Results

Table 2 shows the average rewards for 10 rounds achieved by the random agent, Q-learning agent, and PPO agent. The Q-learning agent achieved an average reward of 326, which was higher than the baseline random agent's average reward of 221. However, the PPO agent significantly outperformed both, achieving an average reward of 967.

Agent	Average Reward
Random	221
Q-learning	326
PPO	967

Table 2: Average Rewards of Different Agents

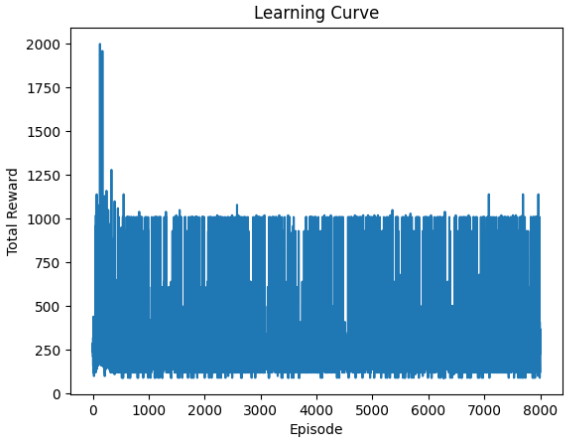

Figure 1: Learning Curve of Q-learning Agent (8000 episodes)

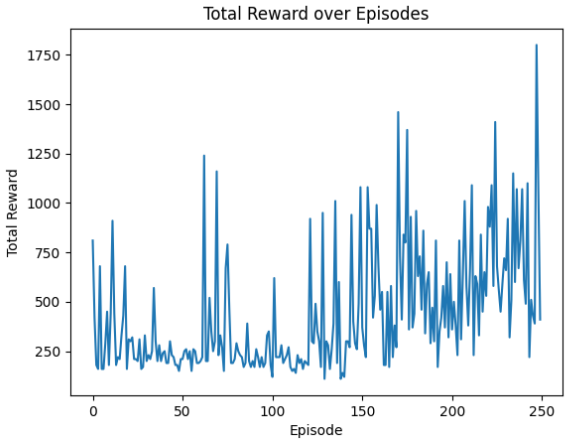

Figure 2: Total Reward over Episodes for PPO Agent (250 episodes)

4.3 Learning Curve Analysis

The learning curve of the Q-learning agent, as shown in Figure 1, indicates significant variability in the rewards obtained throughout the training process. The initial episodes show high fluctuation in rewards, reflecting the agent's exploration phase. As training progresses, the curve does not stabilize, suggesting that the Q-learning agent struggles to consistently learn an optimal policy in the high-dimensional state space of the MsPacman-v5 environment. The rewards often fall below the baseline of 200, highlighting the difficulties faced by the Q-learning algorithm in this complex environment.

In contrast, the learning curve of the PPO agent, depicted in Figure 2, demonstrates a more stable and consistent increase in rewards over the training episodes. The PPO agent quickly surpasses the baseline reward of 200 and continues to improve, achieving an average reward of 967. This stability and improvement suggest that the PPO algorithm is more effective in handling the complexity and dynamics of the MsPacman-v5 environment. The PPO agent's ability to efficiently explore and exploit the environment leads to better performance and higher rewards.

5 Conclusion and Future Work

The Proximal Policy Optimization (PPO) agent demonstrated superior performance and stability in the MsPacman-v5 environment, effectively learning strategies to navigate the complex and dynamic game. In contrast, the Q-learning agent faced significant challenges due to the high-dimensional state space and complex game dynamics, resulting in an average reward below the baseline. These findings suggest that policy-based methods like PPO are better suited for complex environments where efficient exploration and stable learning updates are crucial.

Future work will involve exploring advanced techniques like Deep Q-Networks (DQN) to enhance the performance of value-based methods in complex environments. Additionally, experimenting with different architectures and hyperparameters for PPO could further optimize its performance. Applying the trained agents to other challenging environments will help generalize the findings and evaluate the robustness of the algorithms. Investigating multi-agent scenarios will also provide insights into the interaction dynamics and cooperative behaviors among agents. Finally, transitioning from simulated environments to real-world applications in robotics and automation will test the practical viability of these reinforcement learning methods.

References

[1] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. In *Journal of Artificial Intelligence Research*, volume 47, pages 253–279, 2013.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[3] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 268–277. JMLR. org, 2017.