Investigating the Effectiveness of PPO and DQL Reinforcement Learning Methods on Ms Pacman

Noah Lee, Tom Finzell

Abstract:

This paper investigates two deep reinforcement learning methods on the Atari version of Ms. Pacman–DQL and PPO. Of the two models trained, DQL performed significantly better and was able to learn novel strategies, while the PPO model was unable to learn from the environment effectively, even after hours of training. We conclude that DQL performed better in this relatively simple environment with shorter training, and that further research is needed with greater computational power and time to evaluate the true effectiveness of PPO.

Introduction:

In 2019, OpenAI created a model that played the popular online game Dota 2, beating some of the best players in the world (OpenAI). This revealed to the general public the growing intelligence of machine learning models and their capability to solve problems with no clear correct strategy. The problem of 'solving' Dota 2 involves two main entities–the agent (the player) and the environment (enemies and the Dota 2 map). There is no label to the data (such as in supervised learning), only rewards given to the agent in response to actions given a particular state in the game. This is the problem that reinforcement learning aims to solve.

One popular reinforcement learning method is Q-Learning. This technique creates a table consisting of each possible state and each possible action. For instance, in a 4x4 grid with four possible actions (left, right, up, down), the Q-table will be a 4x16 matrix with each cell representing a Q-value, with the largest Q-value being the action that the agent will take given a specific state (row). When the agent receives a reward, the associated cell on the Q-table is updated, and all cells related to that cell are updated as well. The agent can use epsilon-greedy, a policy that begins training by making random movements (exploration with epsilon = 1) and as the training ends, epsilon decays, and the agent begins using more actions based on the Q-table it has developed through training (exploitation with epsilon = 0). Deep Q-Learning is an extension of Q-Learning, using two deep neural networks–the target and the policy network–to update the Q-table. State-of-the-art Deep Q-Learning (DQL) methods also use convolutional layers in their neural networks to increase generalizability. DQL is an example of the field of Deep Reinforcement Learning (DRL), which are reinforcement learning methods that involve deep neural networks.

Proximal Policy Optimization (PPO) represents a more advanced approach that directly learns a policy (strategy) for action selection rather than estimating action values (Q values). PPO uses an actor-critic architecture where the "actor" network decides what actions to take and the "critic" network evaluates how good those actions were, with both networks learning simultaneously from collected gameplay experience. The key innovation of PPO (from similar networks like REINFORCE) is its "clipped" objective function that prevents the policy from changing too drastically between updates, ensuring stable learning without catastrophic forgetting of successful strategies.

The OpenAI model was compelling due to its use of Proximal Policy Optimization (PPO). OpenAI was able to use incredible compute power to train a model at large scales–allowing the model to train on the

inconceivable number of possible states in a Dota game. To investigate the usefulness and properties of these deep reinforcement learning (DRL) methods, without the incredible compute that is required for more sophisticated games, Pacman was chosen as the ideal candidate to test and implement DRL models on.

Ms. Pacman is an arcade game that serves as an ideal environment for testing deep reinforcement learning algorithms. The game consists of a yellow character (Ms. Pacman) navigating through a maze while collecting dots and avoiding four colored ghosts. The objective is to collect all dots while surviving as long as possible, with power pellets temporarily allowing Ms. Pacman to eat the ghosts for bonus points. The environment presents several key challenges that make it suitable for DRL research: partial observability (ghosts can appear from anywhere), dynamic enemies with different behavioral patterns, sparse and delayed rewards (dots give small immediate rewards, but survival requires long-term planning), and a large state space. Unlike simpler environments like TicTacToe, Ms. Pacman requires the agent to develop sophisticated strategies involving spatial reasoning, temporal planning, and risk assessment. The game lies between simple problems and complex real-world applications, making it an ideal test for comparing different reinforcement learning approaches without requiring the great computational resources needed for games like Dota 2.

Related Work:

This work was greatly inspired by the dissertation of Miguel Diogo Ferraz Araújo of Universidade de Aveiro. Their code on the Pacman environment was used as the foundation for the project and their YouTube video on their results provided an idea for how good these models could be and how long it would take to train them. In Araújo's findings, despite training a Deep Q-Learning model for 16 hours, Pacman inherited somewhat undesirable traits like stalling in the corner instead of continuously going for pellets (Araújo). However, a PPO model trained on 95 hours of continuous gameplay achieved impressive results, enabling it to play and win two maps with four ghosts on high difficulty. These results were not expected to be achieved within my timeframe, but it was hoped that I would be able to get a working Pacman moving on the board that could avoid ghosts and collect pellets to some extent.

OpenAI's research and work in PPO also provided much of the insights as to how to implement the algorithm. Their gymnasium package is used in Araújo's dissertation for the environment for Pacman. In addition, Johnny Code's work on DQL in the simple FrozenLake gymnasium environment also served as a guide on how to implement deep reinforcement models using gymnasium (Johnny Code).

Data:

The data for this project consists of gameplay episodes generated through interaction with the Ms. Pacman environment provided by OpenAI's Gymnasium library (formerly OpenAI Gym). Unlike supervised learning approaches that require pre-labeled datasets, reinforcement learning generates its own training data through trial-and-error interaction with the environment. Each data point represents a single timestep containing a state observation, action taken, reward received, and information about episode termination.

Figure 1 - Screenshot of the Atari Ms. Pacman Environment (Atari ROM)

The state observations are 210x160x3 RGB images representing the current game screen, which are preprocessed for neural network training by converting to grayscale, resizing to 84x84 pixels, and stacking 4 consecutive frames to provide temporal information about movement and direction. This preprocessing reduces computational requirements while preserving essential visual information needed for decision-making. Actions are discrete, consisting of 9 possible movements: no operation, up, down, left, right, and four diagonal directions.

Rewards in the environment follow the original arcade game scoring: +10 points for collecting regular dots, +50 points for power pellets, and progressively higher scores (200, 400, 800, 1600) for eating ghosts after consuming a power pellet. Episodes terminate when Ms. Pacman loses all lives or completes the level, with episodes typically lasting 300-2000 timesteps for untrained agents and potentially 10,000+ timesteps for skilled agents.

The accumulation of training data varies between algorithms. The PPO implementation collects experience in batches of 2048 timesteps before performing policy updates, while the DQL approach stores individual transitions in a replay buffer with capacity for 100,000 experiences, sampling randomly during training. Over the course of training, successful models generate hundreds of thousands to millions of

state-action-reward transitions, with the total dataset size scaling with training duration and episode length as the agent improves.

Methods:

Two deep reinforcement learning algorithms were implemented and compared for the Ms. Pacman environment: Proximal Policy Optimization (PPO) and Deep Q-Learning (DQL). Both utilize convolutional neural networks for visual processing of game frames.

Environment Setup and Preprocessing:

The Ms. Pacman environment from OpenAI's Gymnasium library provides 210x160x3 RGB images as observations, representing the current game screen. To make this suitable for neural network training, a preprocessing pipeline was implemented. The preprocessing begins with frame conversion, where RGB frames are converted to grayscale to reduce compute while preserving essential visual information. Following this, frames are resized from 210x160 to 84x84 pixels, sticking to Atari game standards. Frame stacking is then implemented, where four consecutive frames are stacked together to provide temporal information about movement direction and velocity to pick up game dynamics like ghost movement patterns; this creates a (4,84,84) tensor. Finally, pixel values are normalized to ensure stable neural network training.

The action space consists of 9 discrete actions: no operation, up, down, left, right, and four diagonal movements. The reward structure follows the original arcade game: +10 points for regular dots, +50 for power pellets, and escalating bonuses (200, 400, 800, 1600) for eating ghosts after power pellet consumption.

Proximal Policy Optimization (PPO) Architecture:

The PPO implementation uses an enhanced Actor-Critic architecture specifically designed for the visual complexity of Ms. Pacman:

Convolutional Feature Extractor: A four-layer convolutional network processes the stacked frames:

- Layer 1: 32 filters, 8x8 kernel, stride 4 (captures large-scale maze structure)
- Layer 2: 64 filters, 4x4 kernel, stride 2 (identifies game entities)
- Layer 3: 128 filters, 3x3 kernel, stride 1 (feature detection)
- Layer 4: 128 filters, 3x3 kernel, stride 1 (enhanced feature detection)

Shared Representation Layer: The convolutional output feeds into a two-layer fully connected network (1024→512 units) with dropout regularization (0.2, 0.1) to prevent overfitting to specific maze configurations.

Dual-Head Architecture:

- Actor Head: Two-layer network (512→256→9) outputting action probabilities for policy decisions. The 9 output nodes correspond with the 9 possible actions.
- Critic Head: Two-layer network (512→256→1) estimating state values for advantage computation.

The PPO algorithm implementation includes multiple key components tailored specifically for the Ms. Pacman environment. The clipped surrogate objective prevents destructive policy updates that could cause the agent to forget successful strategies, using a clip ratio of 0.15. Without clipping, a single bad experience such as accidentally running into a ghost could cause the agent to abandon this strategy completely. Generalized Advantage Estimation (GAE) balances the bias-variance trade-off in advantage estimates with λ=0.98, which is crucial for handling the sparse reward structure of Ms. Pacman. The λ=0.98 value means the agent considers both immediate and distant future rewards when evaluating actions. Experience collection gathers 2048 timesteps before each policy update, allowing the agent to experience diverse game situations rather than updating the policy after every single action. Multi-epoch training employs 8 epochs of minibatch updates with a batch size of 256 to maximize data efficiency, similar to a student reviewing the same material multiple times in different chunks to fully understand it (OpenAI).

Deep Q-Learning (DQL) Architecture:
The DQL implementation uses a similar but modified architecture suited for simpler Q-value estimation:
Convolutional Layers: Identical to PPO's feature extractor (32→64→64 filters) but with three layers instead of four, as Q-Learning does not need to learn both a value function and policy like PPO, terminating in a 512-unit fully connected layer that directly outputs Q-values for all 9 actions.

Double DQN Implementation: Two networks are used to prevent overestimation bias:

- Policy Network: For selecting actions
- Target Network: Provides stable Q-value targets that is updated every 5,000 steps

Experience Replay: A circular buffer stores 100,000 state transitions. This allows the agent to learn from a larger pool of past experiences rather than just recent gameplay. This is particularly important for Ms. Pacman where successful strategies must be learned from occasional high-reward episodes.

Exploration Strategy: ε-greedy policy with linear decay from 1.0 to 0.02 over 500,000 steps, preceded by 10,000 warmup steps. This extended exploration phase allows the agent to discover effective pellet collection patterns and ghost avoidance strategies.

Training Procedures:
Both algorithms use identical environment interfaces and evaluation procedures. Training episodes terminate when Ms. Pacman loses all lives or completes the level, with episodes ranging from 300-15,000 timesteps.

<u>PPO Training Cycle:</u>

1. Collect 2048 timesteps of experience
2. Compute advantages using GAE
3. Perform 8 epochs of policy updates - updating both actor and critic network.
4. Update learning rate via exponential decay

<u>DQL Training Cycle:</u>

1. Take action using ε-greedy policy

2. Store experience in replay buffer
3. Sample minibatch and update Q-network every 4 steps
4. Update target network every 5,000 steps

Both implementations include comprehensive logging of training metrics, automatic model checkpointing on performance improvements, and early stopping mechanisms to prevent overtraining. Them hyperparameter choices prioritize training stability over maximum performance (to prevent model explosion), ensuring reproducible results suitable for algorithm comparison and were adjusted over multiple training sessions to improve performance. There is also a script to run three testing episodes and output a video of the sessions.

Training was conducted using Carleton's CS departments' research machines with NVIDIA GPUs.

Results:

The experimental comparison between Deep Q-Learning (DQL) and Proximal Policy Optimization (PPO) on the Ms. Pacman environment revealed significant performance differences between the two algorithms over extended training periods.

Deep Q-Learning Performance

The DQL implementation demonstrated substantial learning progress over the first 2,000 episodes of training. As shown in Figure 1, the agent exhibited clear improvement from initial random performance to sophisticated gameplay strategies. The implementation achieved a maximum reward of 7,440 points with an overall average score of 2,417.1 points across the training period. Recent performance averaged 2,298.5 points with a standard deviation of 909.9 points, achieved through 2.4 hours of training across 7,122 total episodes.

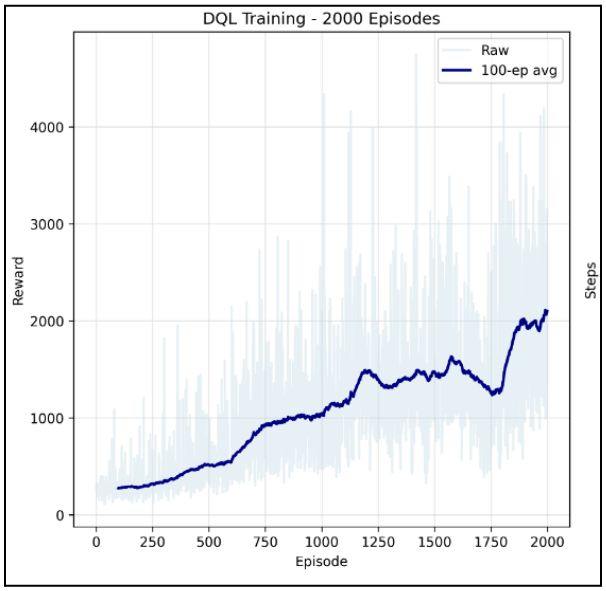

Figure 2 - Plot of first 2000 episode scores for DQL training (only stored 2000 episodes because storing this with the experience replay was causing the machine to crash)

The learning curve shows three distinct phases: an initial exploration period (episodes 0-500) where scores remained below 500 points, a rapid improvement phase (episodes 500-1500) with steadily increasing performance, and a stabilization period (episodes 1500-2000) where the agent achieved consistent scores above 2,000 points. The final epsilon value of 0.02 indicates the agent had transitioned from exploration to exploitation, relying primarily on learned Q-values for decision-making. Following 2,000 episodes (not shown above), the agent reached an average score of 2,400 points by episode 7,000 and plateaued in performance.

Video analysis revealed that the trained DQL agent successfully learned fundamental Ms. Pacman strategies, including dot collection, basic ghost avoidance, and maze navigation. However, performance variability (standard deviation of 909.9) suggests the agent occasionally encountered unfamiliar game states where its learned policy became suboptimal, leading to premature episode termination.

Proximal Policy Optimization Performance

In contrast, the PPO implementation failed to demonstrate meaningful learning progress despite long training (7 hours). As illustrated in Figure 2, the PPO agent's performance remained flatlined around 200-250 points throughout 15,002 episodes of training, showing no significant improvement over the baseline random policy (which is what DQL starts off with at 200 points). The maximum reward achieved was approximately 2,000 points, though these were rare outliers. The average performance plateaued at roughly 250 points throughout training, with the agent following a single path regardless of the ghost movements and getting stuck until a ghost eats it.

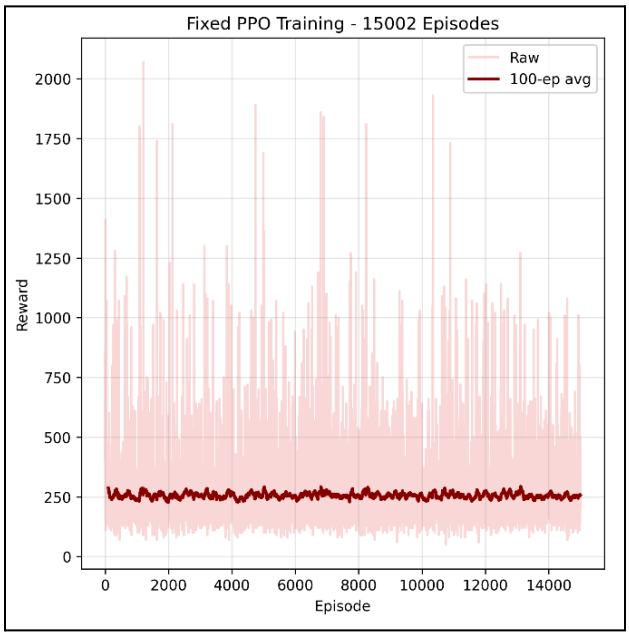

Figure 3 - Plot of all episode scores for PPO training

The PPO learning curve exhibits high variance with no upward trend, indicating the agent failed to discover effective gameplay strategies. Observational analysis revealed the agent developed a fixed behavioral pattern of following the same route repeatedly until encountering a wall, where it would

remain stationary until eliminated by ghosts. This suggests the PPO implementation may have suffered from premature convergence to a poor local optimum or inadequate exploration of the action space.

Comparative Analysis
The results demonstrate a clear superiority of the DQL approach for this specific implementation of Ms. Pacman. In terms of learning efficiency, DQL achieved meaningful improvement within 2,000 episodes, while PPO showed no progress across more than 15,000 episodes. The score performance differential was particularly striking, with DQL's average score of 2,417 points being nearly ten times higher than PPO's plateau of approximately 250 points. Most importantly, DQL developed strategic behaviors including adaptive dot collection and ghost avoidance patterns, while PPO only developed rigid and unsuccessful movement patterns that never progressed beyond basic responses.

Performance Context
These results align with previous research findings where DQL achieved reasonable performance on Atari games, though they fall short of the sophisticated gameplay demonstrated in Miguel Diogo Ferraz Araújo's dissertation, where a PPO model trained for 95 hours achieved expert-level performance. The discrepancy suggests that successful PPO implementation for Ms. Pacman may require significantly longer training periods, different hyperparameter configurations, or architectural modifications not explored in this study.

The DQL agent's ability to achieve high scores demonstrates successful learning of core Pacman mechanics, representing a meaningful step toward competent gameplay despite not reaching human expert levels or the levels of previous studies.

Discussion:
Several limitations restrict this investigation. The failure of the PPO implementation may reflect suboptimal hyperparameter choices, insufficient training duration, architectural limitations, or even computational limitations, rather than fundamental algorithmic inadequacy, as evidenced by successful PPO implementations in prior research that required longer training periods with more computational power. Additionally, the study's focus on a single game environment and single map limits broader conclusions about algorithm performance across general reinforcement learning problems. The hyperparameter selections prioritized training stability over maximum performance (as the machines routinely crashed or led to Q-value explosions), potentially hampering both algorithms' learning potential.

Future research should investigate extended PPO training periods beyond 15,000 episodes, systematic hyperparameter optimization using techniques like grid search or Bayesian optimization, and architectural modifications such as attention mechanisms or improved reward shaping. Comparative evaluation across multiple Atari environments would provide more robust algorithmic insights, while investigating more sophisticated Neural Network implementations of DQN and PPO could yield superior performance. Furthermore, analyzing the learned representations in this study and finding out what exactly is being captured in the convolutional layers may give an idea of whether our networks are capturing meaningful patterns in the gameplay.

Given the small scope of this problem, this specific investigation did not pose any ethical concerns. However, reinforcement learning methods are increasingly being used in the real world such as in computational adaptive tests and self-driving cars, and the consequences of the actions those models have (on children and on roads) are something to be weary of.

References:

1) OpenAI. "OpenAI Five Defeats Dota 2 World Champions." *Openai.com*, 2019, openai.com/index/openai-five-defeats-dota-2-world-champions/.

2) Araújo, Miguel Diogo Ferraz. *Machine Learning Agents for Computer Games*. 2021. Accessed 6 June 2025.

3) Johnny Code. "Deep Q-Learning/Deep Q-Network (DQN) Explained | Python Pytorch Deep Reinforcement Learning." *YouTube*, 20 Dec. 2023, www.youtube.com/watch?v=EUrWGTCGzlA. Accessed 9 Sept. 2024.