# COMP 579 Final Project Report: Reinforcement Learning for Ms. Pac-Man

**Ivy Hu**[1] **Simon Li**[2] **Kenza Bellebouir**[1]
nanqing.hu@mail.mcgill.ca xi.yang.li@mcgill.ca kenza.bellebouir@mail.mcgill.ca
[1]Department of Computer Science, McGill University
[2]Department of Electrical Engineering, McGill University

## Abstract

We investigate the comparative performance of Rainbow DQN and Proximal Policy Optimization (PPO) in the visually complex Ms. Pac-Man environment. Our study evaluates learning dynamics, hyperparameter sensitivity, and robustness to visual perturbations. Rainbow DQN consistently achieves higher peak rewards and demonstrates stronger long-term learning, while PPO shows faster early convergence but suffers from instability. We also conduct ablation studies and color-based generalization tests, offering practical insights into the role of architecture and training design in sparse-reward, high-variance domains.

## 1 Introduction

Deep reinforcement learning (RL) has achieved notable success in high-dimensional decision-making tasks, particularly within the Arcade Learning Environment (ALE) (1), which provides a unified platform for evaluating algorithms on Atari 2600 games (3). These environments are characterized by partial observability, sparse and delayed rewards, and the need for both short-term reactivity and long-term planning.

Among them, *Ms. Pac-Man* is a particularly challenging domain due to its stochastic dynamics, diverse strategic possibilities, and visually rich inputs. It is thus well-suited for evaluating algorithmic robustness and scalability. In this work, we compare two prominent deep RL methods: Rainbow DQN (2), an enhanced value-based agent that integrates six key improvements over DQN, and Proximal Policy Optimization (PPO) (5), a widely used policy-gradient algorithm known for its simplicity and stability.

We implement both agents in the `MsPacman-v0` environment and conduct a detailed empirical evaluation. Beyond comparing baseline performance, we perform an ablation study of Rainbow DQN to assess the contribution of prioritized experience replay, noisy networks, and multi-step returns. To evaluate generalization, we test trained agents under visual domain shifts introduced via color perturbations. We further analyze sensitivity to hyperparameters by conducting targeted sweeps to identify configurations yielding strong and stable performance.

This study provides insights into the design and evaluation of RL algorithms in complex visual settings, highlighting the internal dynamics and generalization capabilities of Rainbow DQN relative to PPO.

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

## 2 Background

### 2.1 Reinforcement Learning and MDPs

Reinforcement Learning formalizes sequential decision-making as a Markov Decision Process (MDP), defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{P}(s'|s, a)$ the transition model, $\mathcal{R}(s, a)$ the reward function, and $\gamma \in [0, 1)$ the discount factor. The agent selects actions to maximize expected return $\mathbb{E}[\sum_t \gamma^t r_t]$ based on its interactions with the environment.

### 2.2 The Ms. Pac-Man Environment

The `MsPacman-v0` environment is a high-dimensional, partially observable MDP. Each observation is a $210 \times 160 \times 3$ RGB frame, preprocessed to $84 \times 84$ grayscale. The action space consists of discrete directional commands and a no-op. Reward signals are sparse, with +10 for pellets, +200 for ghosts after consuming a power pellet, and a time-step penalty of –1. An episode ends when the agent loses three lives.

This environment presents multiple challenges: pixel-based observations limit access to the full game state, ghost behavior is stochastic and partially unpredictable, and the sparse reward structure complicates credit assignment and learning signal propagation. Additionally, the lack of intermediate rewards increases the difficulty of learning effective threat avoidance or path planning strategies.

### 2.3 Rainbow DQN

Rainbow DQN (2) combines several enhancements to the original DQN (3), aiming to improve sample efficiency, stability, and exploration. Double Q-learning addresses overestimation bias by decoupling action selection and evaluation in target computation. The dueling network architecture separates state-value and advantage estimation to refine action selection. Prioritized experience replay focuses learning on high-error transitions, improving data efficiency. Noisy networks inject trainable stochasticity into parameters, facilitating more efficient exploration. Multi-step returns accelerate learning by propagating rewards over multiple future steps. Distributional RL (C51) models a categorical distribution over future returns, enabling richer value representation.

### 2.4 Proximal Policy Optimization (PPO)

PPO (5) is a first-order policy-gradient algorithm that improves training stability by limiting policy updates through a clipped surrogate objective:

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

where $r_t(\theta)$ is the probability ratio between the new and old policies, and $\hat{A}_t$ is an estimate of the advantage function. PPO strikes a balance between exploration and stability, making it suitable for a wide range of tasks without extensive tuning.

### 2.5 Evaluation Challenges

Evaluating agents in visually rich environments like Ms. Pac-Man is non-trivial. RL training is often unstable due to non-stationarity, bootstrapping errors, and high sensitivity to hyperparameters. Sample efficiency is critical, especially when training from pixel inputs requires millions of frames. Generalization remains an open challenge, as agents frequently overfit to the visual statistics of the training domain. Even minor pixel-level perturbations can lead to catastrophic performance drops, as shown in (7).

### 2.6 Related Work

Prior approaches to Ms. Pac-Man have explored architectural and algorithmic enhancements. Toromanoff et al. (6) combined CNNs with recurrent networks to mitigate partial observability, achieving performance significantly better than DQN. Other works, such as Pieters et al. (4), introduced planning-based methods like Monte Carlo Tree Search for improved control, though these typically

rely on engineered features. In contrast, we adopt a pure pixel-based learning paradigm and extend the robustness analysis by introducing controlled visual perturbations during evaluation.

# 3 Methodology

## 3.1 Environment

We evaluate agents in the `MsPacman-v0` environment from OpenAI Gym, based on the Arcade Learning Environment (ALE) (1). The environment outputs raw $210 \times 160 \times 3$ RGB frames, which we preprocess into $84 \times 84$ grayscale or color-normalized images scaled to $[0, 1]$. To encode temporal dynamics, we stack the last 4 frames as input. The action space comprises 9 discrete actions: four cardinal, four diagonal directions, and a no-op. Rewards are sparse: +10 for pellets, +200 for ghosts after a power pellet, and –1 per time step. Episodes end after the agent loses all 3 lives.

## 3.2 Preprocessing

We employ two preprocessing pipelines:

**Baseline:** RGB frames are converted to grayscale, resized to $84 \times 84$ via bilinear interpolation, normalized, and stacked across 4 frames.

**Color Perturbation:** To assess robustness, we introduce a custom `ColorPreprocessFrame` wrapper. Frames are first grayscaled, then tinted or transformed using OpenCV colormaps (e.g., `cv2.COLORMAP_JET`). Tints include red, green, and blue channels. The result is a 3-channel RGB image that preserves semantic content while altering visual appearance.

## 3.3 Model Architectures

**Rainbow DQN.** We implement the full Rainbow DQN (2), incorporating: a convolutional encoder (32, 64, 64 filters); a dueling head with separate value and advantage streams; NoisyLinear layers ($\sigma = 0.017$); a distributional output with 51 atoms over $[-10, 10]$; prioritized replay ($\alpha = 0.6$, $\beta$ annealed from 0.4 to 1.0); and multi-step returns with $n = 3$. The target network is updated every 1,000 steps. Training uses the Adam optimizer with a learning rate of $1 \times 10^{-4}$.

**Proximal Policy Optimization (PPO).** PPO follows a shared convolutional encoder (identical to Rainbow), feeding into a 512-unit fully connected layer, then branching into actor and critic heads. We optimize the clipped surrogate objective ($\epsilon = 0.1$) and use GAE with $\lambda = 0.95$. PPO is trained with a batch size of 32, a learning rate of $2.5 \times 10^{-4}$, across 1000 updates with 256-step rollouts (256,000 environment steps in total).

## 3.4 Training Details

Agents receive frame-stacked inputs and are trained using Adam. Rainbow DQN is trained for 30,000–50,000 frames, while PPO trains for 1,000 steps due to its on-policy nature. A fixed seed (357) is used across all experiments. Evaluation follows each training run using the same configuration, but without exploration noise.

Experiments were conducted on an NVIDIA RTX 3080 laptop (ablation and color studies) and a Mac notebook (hyperparameter tuning), with training durations ranging from 2–5 hours depending on the task.

## 3.5 Evaluation Protocol

Each agent is evaluated using a fixed frame budget. We report: average episode return over training and final episode return post-training, as visualized in the results.

## 3.6 Hyperparameter Ablation

We conducted a grid search on three key Rainbow DQN hyperparameters: Adam learning rate $\eta$, multi-step return length $n$, and replay prioritization exponent $\alpha$, with all other settings fixed as

in Section 3.4 ($\gamma = 0.99$, buffer size $10^5$, batch size 32, target update every 1,000 frames, and $\beta$ annealed from 0.4 to 1.0 over 100,000 frames).

Specifically, we evaluated:

$$\eta \in \{1 \times 10^{-5}, 5 \times 10^{-5}\}, \quad n \in \{3, 5\}, \quad \alpha \in \{0.4, 0.6\},$$

yielding eight combinations, each trained for 30,000 frames using a fixed seed. This preliminary sweep identified settings that promoted faster reward growth and highlighted configurations with higher variance.

# 4 Results and Discussion

## 4.1 PPO Hyperparameter Sensitivity

We evaluated three PPO configurations: *Balanced* (blue), *Fast Training* (orange), and *Aggressive* (green). The Aggressive setup, combining long rollouts, tight gradient clipping, and low entropy, consistently achieved the highest rewards (approaching 1,000), demonstrating a favorable trade-off between stability and efficiency. In contrast, Fast Training plateaued early at lower scores, while Balanced learning remained steady but suboptimal. Results were reproducible across three runs for each configuration.

## 4.2 Comparative Performance: Rainbow DQN vs. PPO

Figure 2 compares PPO and Rainbow DQN learning in `MsPacman-v0`. PPO displays early convergence to a moderate range (600–800 points) but deteriorates after update 800, likely due to sampling noise and hyperparameter sensitivity. It still manages to acquire basic survival and collection behaviors.

Rainbow DQN, though more volatile, shows a clearer upward trajectory with multiple peaks surpassing 1,000 and 2,000 points. These spikes indicate discovery of higher-reward strategies, benefiting from Rainbow's architectural features like prioritized replay and distributional outputs. Despite its instability, Rainbow demonstrates superior long-term reward potential within the same frame budget.

## 4.3 Insights and Interpretation

Rainbow DQN consistently outperforms PPO in peak performance, affirming the effectiveness of value-based methods enhanced with multi-step returns, noisy exploration, and replay prioritization in sparse-reward, long-horizon tasks. However, high variance in both methods suggests neither has fully stabilized. PPO's reward collapse highlights the challenges of on-policy learning, while Rainbow's fluctuations point to sensitivity in replay dynamics and learning rate tuning. Rainbow's stronger trend suggests greater promise under the current constraints, while PPO may benefit from longer rollouts or adjusted GAE parameters.

## 4.4 Rainbow DQN Hyperparameter Ablation

Figure 3 presents eight Rainbow DQN configurations varying in learning rate $\eta$, multi-step return $n$, and prioritization strength $\alpha$. Four settings stand out: D, E, G, and H — all using $\eta = 5 \times 10^{-5}$ or compensating via increased $\alpha$, and three with $n = 5$. These combinations yielded higher final rewards, suggesting that deeper bootstrapping and moderately aggressive learning rates facilitate better performance in sparse-reward environments.

## 4.5 Extended Sweep: Replay and Update Scaling

To test broader settings, we re-ran all eight ablations with a larger replay buffer (1M), slower target updates (every 5,000 steps), and extended $\beta$-annealing (500K frames). As shown in Figure 4, all configurations improved relative to their 30K-frame baselines, with C_large, E_large, G_large, and H_large showing the greatest gains. These settings share a larger buffer and, in most cases, a 5-step return.

Despite improvements, all runs remained highly variable—fluctuating by $\pm 500$ points per episode. This reflects the stochastic nature of prioritized replay and the challenge of sparse rewards in Ms. Pac-Man. While extended settings enhanced median returns, further progress may require additional seeds, longer training durations, or targeted variance-reduction techniques.

### 4.6 Seed-Level Reproducibility on Rainbow DQN

To assess the stability of our hyperparameter choices, we re-ran Configurations A and E across three different random seeds $(0, 1, 2)$. Figures 5 and 6 show, for each seed, the episode-wise total reward (dashed lines), the mean curve (solid black), and the shaded $\pm 1\sigma$ band.

Overall, Configuration E achieves a higher average reward than A, but at the cost of increased jitter. These results underscore the importance of reporting across multiple seeds—averaging over 3–5 runs provides a more reliable estimate of true performance than any single trial. Given the wide per seed variability, reporting a single run can be misleading. Averaging across 3–5 seeds provides a more robust estimate of true algorithmic performance.

### 4.7 Rainbow DQN Ablation Study

To elucidate the contributions of individual components in the Rainbow DQN architecture, we conducted an ablation study comparing three simplified variants: **nstep1**, with multi-step returns disabled by setting $n = 1$; **no_prior**, using uniform sampling in place of prioritized replay; and **no_noisy**, where NoisyLinear layers are replaced with standard linear ones, disabling learned exploration.

Figure 7 presents training performance across 45 episodes. All variants exhibit similar average scores (200–700 points), yet distinct patterns emerge.

The **full Rainbow** agent occasionally achieves notably higher rewards—e.g., sharp peaks around episodes 7 and 20—implying that the integration of prioritized replay, multi-step returns, and noisy exploration enhances strategic discovery. Nonetheless, its performance remains volatile in early training, suggesting instability.

The **no_prior** agent performs competitively in later stages, at times surpassing full Rainbow after episode 25. This implies that prioritized replay, while beneficial for early learning, may not be crucial for final policy quality under constrained training.

The **no_noisy** variant shows pronounced late-stage gains, exceeding 1000 points in several episodes. This suggests that deterministic policies may yield greater stability in this domain, albeit with less effective initial exploration.

By contrast, the **nstep1** configuration consistently underperforms, underscoring the critical role of multi-step bootstrapping in environments with sparse, delayed rewards such as Ms. Pac-Man.

## 5 Conclusion

In the `MsPacman-v0` environment, Rainbow DQN outperformed PPO in terms of peak reward and long-term potential. While both agents exhibited early learning, Rainbow's combination of prioritized replay, multi-step returns, and noisy exploration proved more effective in sparse-reward, long-horizon scenarios, despite occasional instability.

PPO showed greater initial stability but suffered from later performance collapse, underscoring its sensitivity to on-policy sampling and hyperparameters. Rainbow's variance highlights its own limitations, yet in our frame budget, it demonstrated superior sample efficiency and strategic discovery.

Future work could enhance both agents by incorporating memory (e.g., LSTM) for improved temporal reasoning, extending training to tens of millions of frames to approach superhuman performance, and doing multiple runs per setting to mitigate training variance. These would strengthen agent robustness and further illuminate the challenges of mastering complex environments like Ms. Pac-Man.

# A Appendix

## A.1 PPO Hyperparameter Sweeps

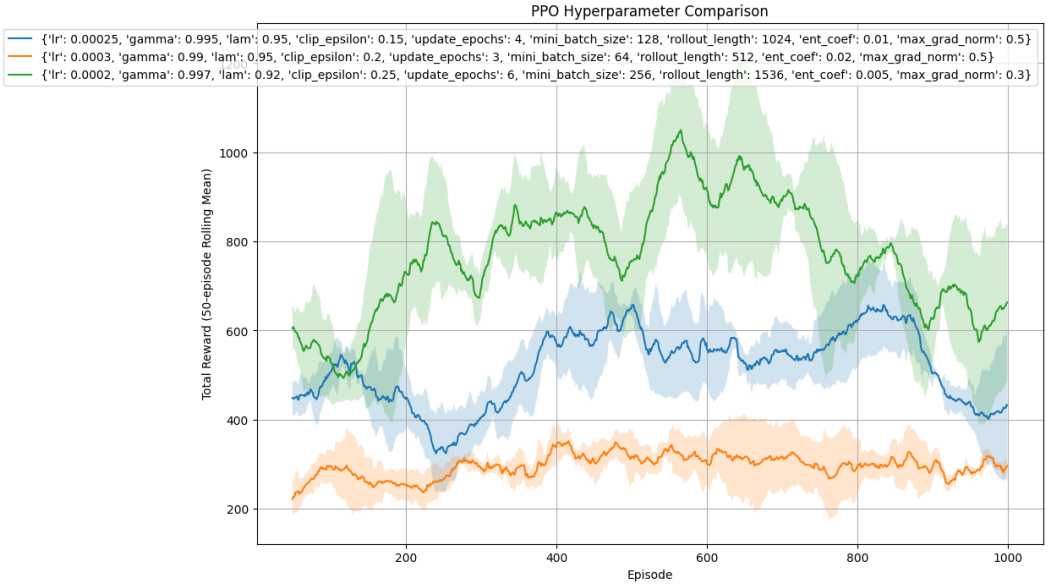


Figure 1: PPO Hyperparameters Optimization Results

## A.2 Rainbow DQN vs PPO

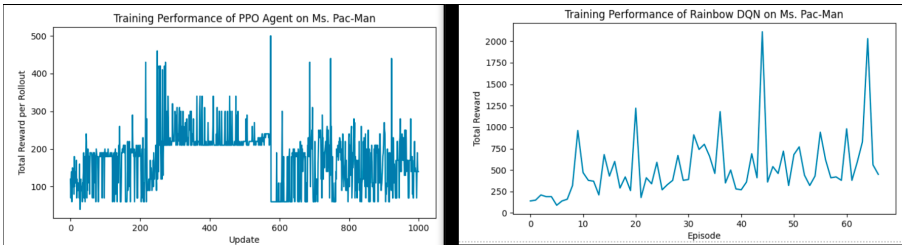


Figure 2: Training performance of PPO and Rainbow DQN agents.

## A.3 Rainbow DQN Ablation Results

## A.4 Extended Hyperparameter Sweep

## A.5 Seed-Level Reproducibility

## A.6 Ablated Rainbow DQN Variants

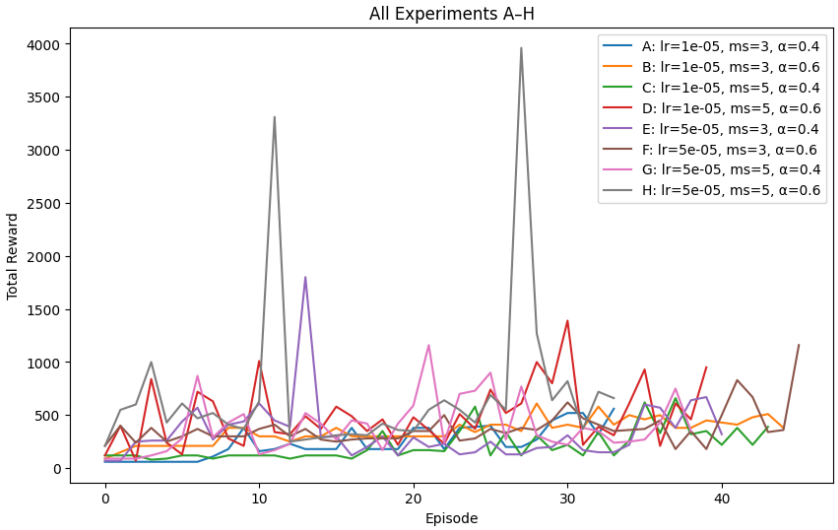


Figure 3: Learning curves for the eight hyperparameter configurations: $\eta \in \{1 \times 10^{-5}, 5 \times 10^{-5}\}$, $n \in \{3, 5\}$, $\alpha \in \{0.4, 0.6\}$.

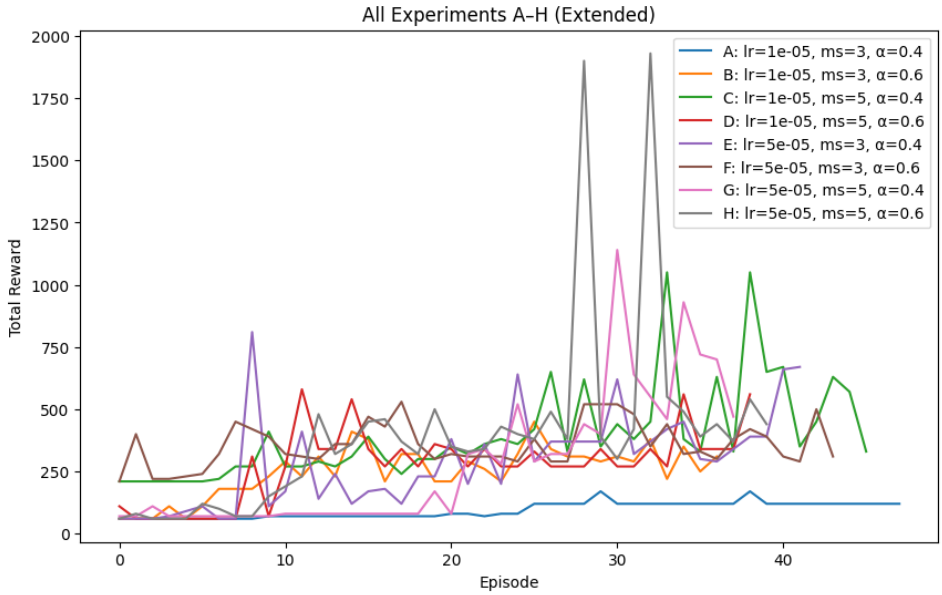


Figure 4: Learning curves for the eight "large" hyperparameter runs (A_large–H_large).

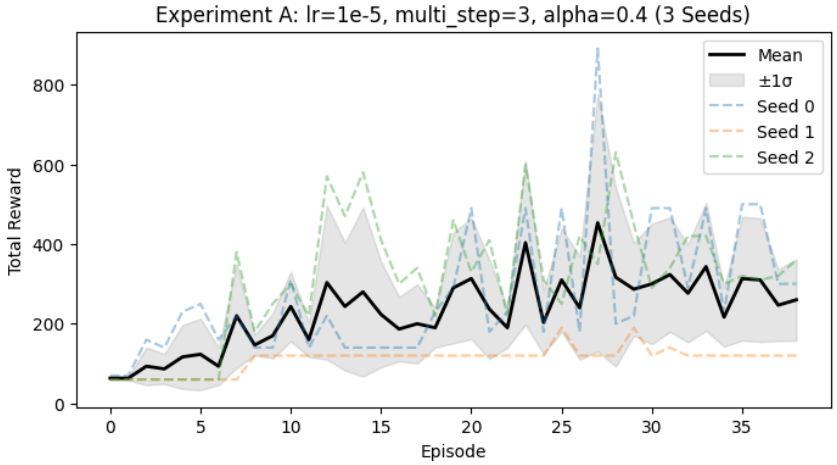


Figure 5: Configuration A ($\eta = 1 \times 10^{-5}$, $n = 3$, $\alpha = 0.4$) across three seeds.

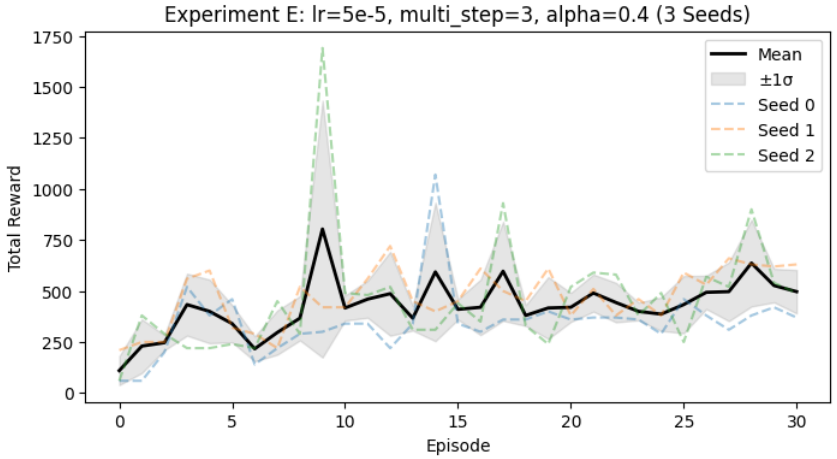


Figure 6: Configuration E ($\eta = 5 \times 10^{-5}$, $n = 3$, $\alpha = 0.4$) across three seeds.

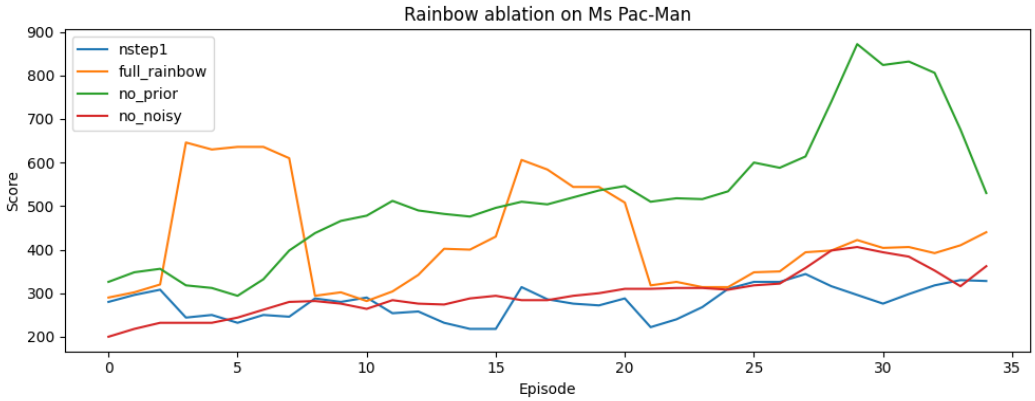


Figure 7: Training performance of Rainbow DQN and ablated variants on Ms. Pac-Man.

## References


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

[2] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad G Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[3] 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.

[4] Roel Pieters and Sander Bakkes. Monte carlo tree search for the ms. pac-man competition. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2016.

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

[6] Mikael Toromanoff, Alessandro Lazaric, and Olivier Pietquin. Deep reinforcement learning with attention for slate markov decision processes with high-dimensional states and actions. In *International Conference on Machine Learning (ICML)*, 2019.

[7] Amy Zhang, Andrew Ballard, Tom Zahavy, Huan Xu, Joelle Pineau, and Marc G Bellemare. An investigation of model-free planning. In *International Conference on Learning Representations (ICLR)*, 2020.