Comparative Analysis of Reinforcement Learning Algorithms: Proximal Policy Optimization, Prioritized Experienced Repay, Deep Q-Network and Double Deep Q- Network in the Lunar Lander Simulation

Balamugunthan Ramesh
Department of Computer Science
University of Bath
Bath, BA2 7AY
bmr40@bath.ac.uk

Deepu Raneesh
Department of Computer Science
University of Bath
Bath, BA2 7AY
dr833@bath.ac.uk

Aditya Sanjeev Kamte
Department of Computer Science
University of Bath
Bath, BA2 7AY
ask206@bath.ac.uk

Nitish Kumar Jha
Department of Computer Science
University of Bath
Bath, BA2 7AY
nj585@bath.ac.uk

Mohammad Rayyan Khan
Department of Computer Science
University of Bath
Bath, BA2 7AY
mrk52@bath.ac.uk

1 Introduction

According to Wiering and Van Otterlo (2012), reinforcement learning (RL) is a machine learning method where an agent learns to make decisions by interacting with its environment, similar to natural learning processes that maximize rewards through trial and error. The Lunar Lander simulation exemplifies this, requiring a spaceship to land safely on the moon, balancing fuel use and different objectives (Brady and Paschall, 2010).

We applied four advanced RL techniques namely Deep Q-Network (DQN), Double Deep Q-Network (DDQN), Prioritized Experience Replay (PER), and Proximal Policy Optimization (PPO) (Fan et al., 2020) to tackle this challenge. PER improves learning efficiency by prioritizing significant experiences, while PPO ensures stability in environments requiring continuous action. DDQN, improving on DQN, reduces bias by using a dual-network architecture. These methods were chosen for their efficiency in complex RL tasks, and this study assesses their performance through learning curves and success rates to identify the best solution for the Lunar Lander problem and offer insights for other RL applications.

2 Problem Definition

The "Lunar Lander" simulation, recognized as a benchmark challenge in the domain of reinforcement learning, presents the complex task of autonomously navigating a spacecraft to achieve a soft landing on the moon's surface (Brady and Paschall, 2010). This simulation intricately models the spacecraft's dynamics, where the agent must adeptly manage its descent using thrusters to ensure it lands within a designated target area, minimizing fuel use and avoiding detrimental impacts. The task is characterized by a continuous state space, including variables such as position, velocity, orientation, and angular speed, and a continuous action space encompassing possible thrust actions.

States: The state of the lunar lander is defined by eight continuous variables: horizontal position, vertical position, horizontal velocity, vertical velocity, angle, angular velocity, and two boolean touch sensors indicating contact with the ground.

Actions: The action space consists of discrete choices: do nothing, fire the left orientation thruster, fire the main engine, or fire the right orientation thruster. Each action directly influences the lander's movement and orientation.

Transition Dynamics: The lander's state transitions are governed by the physics of motion under gravity, inertia, and the application of thrust. These dynamics are complex due to the non-linear interactions between actions and the resulting state transitions.

Reward Function: The reward function is designed to encourage safe landing by providing positive rewards for decreasing distance to the landing pad, successful landing sequences, and conserving fuel, with penalties for crashing or moving away from the target (Sutton and Barto, 2018).

Problem Difficulty: The "Lunar Lander" poses substantial challenges for reinforcement learning due to its high-dimensional and continuous state and action spaces, which complicate the learning process. The environment also includes stochastic elements, such as variability in landing surface and initial conditions, increasing the complexity of achieving consistent performance.

Unsuitability for Tabular Methods: Because of the "Lunar Lander" simulation's continuous state and action spaces, tabular reinforcement learning approaches are inappropriate, making discrete methods unfeasible and computationally demanding (Tavares and Chaimowicz, 2018). Thus, in order to compare four sophisticated algorithms for reinforcement learning, this study examines Proximal Policy Optimisation (PPO),Prioritized Experience Replay (PER), Double Deep Q-Network (DDQN), and Deep Q-Network (DQN), each of which offers a different approach to learning and making decisions (Schulman et al., 2015)(Van Hasselt, Guez and Silver, 2016)(Lillicrap et al., 2015)(Mahmood et al., 2018). In order to determine the best strategy for accomplishing successful and efficient landings, the study will examine learning curves, efficiency, and overall performance. By emphasising accuracy and flexibility, this study not only advances scholarly knowledge but also directs the use of reinforcement learning techniques in autonomous systems and robotics in the actual world.

The study seeks to answer the following questions:

- How do the learning curves of PPO, PER, DQN and Double DQN differ in the context of the Lunar Lander simulation?
- Which algorithm demonstrates the most consistent performance in terms of landing success and fuel efficiency?
- What are the practical implications of each algorithm's performance for real-world applications requiring autonomous control and decision-making?
- This investigation will not only contribute to the academic understanding of reinforcement learning algorithms but also guide the selection of appropriate techniques for similar real-world applications in autonomous systems and robotics.

3 Background

Originally inspired by video games, the Lunar Lander simulation in the OpenAI Gym toolkit challenges an agent to safely land a spacecraft on a designated pad. This environment is exemplary of complex control problems, demanding precise handling of fuel, speed, and angle of descent, thereby mirroring real-world issues in autonomous navigation and control systems.

3.1 Deep Q-Network and Double Deep Q-Network

Deep Q-Networks (DQN) have been pivotal in handling discrete action spaces, like those in Lunar Lander, using deep neural networks. However, DQN struggles with continuous actions, needing

substantial resources for convergence. Double Deep Q-Networks (DDQN) address this by decoupling action selection from value estimation, mitigating overestimation bias. This improvement enhances stability and reliability, demonstrating superior performance in environments similar to Lunar Lander. DDQN's utilization of two networks for value estimation and action selection results in more accurate Q-value estimates, making it a compelling choice for reinforcement learning in discrete action environments.(Van Hasselt, Guez and Silver, 2016)

3.2 Policy Gradient Methods (e.g., REINFORCE, PPO)

Policy Gradient methods are suitable for continuous action spaces as they optimize the policy directly, facilitating a more nuanced exploration-exploitation balance. They are particularly effective in environments where action spaces are not easily discretized. However, these methods are susceptible to high variance and demand meticulous tuning of parameters like learning rates and discount factors. Proximal Policy Optimization (PPO) has gained traction for its robust performance and user-friendly approach, especially in simulations requiring both discrete and continuous actions(Sutton et al., 1999).

3.3 Prioritized Experience Replay (PER)

Prioritized Experience Replay modifies the traditional approach by changing how transitions are sampled from the replay buffer. Instead of sampling uniformly from the buffer, PER samples transitions with probabilities relative to their importance, which is usually measured by the magnitude of their temporal difference (TD) error (Schaul et al., 2015).

4 Method

This section outlines the reinforcement learning (RL) methods applied to solve the decision-making challenges presented in the "Lunar Lander" environment, as provided by OpenAI Gym. The environment simulates the task of landing a spacecraft, requiring precise control and strategic planning. The RL techniques discussed here include Proximal Policy Optimization (PPO), PriortizationExperienceReplay(PER), Deep Q-Network (DQN), and Double Deep Q-Network (DDQN), with a particular emphasis on PPO due to its efficiency and implementation flexibility.

4.1 Proximal Policy Optimization (PPO):

PPO is a variant of policy gradient methods. It optimizes a special objective function that includes a clipping mechanism to moderate updates, ensuring that the new policy does not deviate too drastically from the old one. This feature prevents harmful large updates that could destabilize the learning process. The method uses two important techniques: it limits the size of policy updates using a clipping parameter and it employs multiple epochs of stochastic gradient descent to optimize the policy using mini-batches of data (Mohammad Hasani Zade and Mansouri, 2021). For more information refer to Appendix B.

The advantage $A(s, a)$ is estimated as the difference between the observed return R and the estimated state value $V(s)$:

$$A(s, a) = R - V(s)$$

The policy ratio $\rho(\theta)$ compares the new policy $\pi(a|s, \theta_{\text{new}})$ with the old policy $\pi(a|s, \theta_{\text{old}})$:

$$\rho(\theta) = \frac{\pi(a|s, \theta_{\text{new}})}{\pi(a|s, \theta_{\text{old}})}$$

The PPO loss function aims to maximize the clipped surrogate objective:

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

Where ϵ is a hyperparameter that constrains the policy ratio $\rho(\theta)$ to lie within the interval $[1 - \epsilon, 1 + \epsilon]$. This clipping prevents the new policy from deviating too far from the old policy.For more information please refer to Appendix A(8.1) and B.

Rationale for Selection:
PPO was chosen primarily for its balance between performance and ease of implementation. It avoids the complexity of computing second-order derivatives, unlike some of its predecessors like TRPO, making it computationally less intensive and simpler to implement. Moreover, its resistance to hyperparameter changes provides robustness, making it suitable for the dynamically challenging "Lunar Lander" scenario. (Schulman et al., 2017)

4.2 Prioritized Experience Replay

Unlike traditional methods that sample experiences uniformly, PER prioritizes important experiences so that they are replayed more frequently. The importance of an experience is measured based on its temporal difference (TD) error, which quantifies how surprising or unexpected the outcome of an action was given the current state. Core components of PER are Priority Calculation, Sampling Probability, and Importance Sampling Weights(Schaul et al., 2015). For more information please refer to Appendix A(8.1)and B.

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

The priority of transition i, denoted p_i, is defined as $p_i = |\delta_i| + \epsilon$, where:

- $|\delta_i|$ is the absolute value of the TD error for transition i,
- ϵ is a small positive constant to ensure no transition has a zero probability of being chosen,
- α is a parameter that controls the degree of prioritization, with $\alpha = 0$ leading to uniform sampling.

4.3 Deep Q-Network (DQN):

DQN integrates classical Q-learning with deep learning by using a neural network to approximate the Q-value function. This network predicts the reward expected from each possible action at a given state. Stability is achieved through techniques like experience replay and fixed Q-targets, which mitigate the issues arising from correlated data and moving targets in the learning updates (Li et al., 2022). For more information please refer to Appendix A(8.1)

In DQN, the loss function is defined to minimize the difference between the currently estimated Q-values and the target Q-values, which are calculated based on the Bellman equation. For more information refer to Appendix B. The loss is computed as the Mean Squared Error (MSE) between these two values:

$$L = \frac{1}{N} \sum (y_i - Q(S, A; \theta))^2$$

Where:

- $y_i = R + \gamma \max_{a'} Q(S', a'; \theta^-)$ for the target network with parameters θ^-.
- $Q(S, A; \theta)$ is the predicted Q-value from the current network with parameters θ.
- N is the number of samples in the batch.(Mnih et al., 2013)

4.4 Double Deep Q-Network (DDQN):

DDQN improves upon the traditional DQN by decoupling the action selection from the value estimation, significantly reducing the overestimation bias seen in DQN. It also introduces 'soft updates' to gradually integrate changes into the target network, enhancing the stability of the learning updates (Van Hasselt, Guez and Silver, 2016). For more information please refer to Appendix A(8.1). DDQN addresses the overestimation of Q-values by decoupling the selection and evaluation of the action in the Q-value update:

$$y_i = R + \gamma Q(S', \arg\max_{a'} Q(S', a'; \theta); \theta^-)$$

Here, the action a' that maximizes the Q-value in the next state S' according to the current network θ is used, but the value of that action is estimated using the target network θ^-, reducing overestimating (Van Hasselt, Guez and Silver, 2016). For more information refer to Appendix B.

5 Results

We assess Proximal Policy Optimization (PPO), Deep Q-Network (DQN), Double Deep Q-Network (DDQN), and DQN enhanced with Prioritized Experience Replay (Priority). These agents were compared against informative baselines to illustrate their learning efficiency and policy improvement over training episodes. The agents were trained to control a simulated lunar lander module with the objective of landing it safely. The performance of each agent was measured in terms of cumulative rewards per episode, with higher rewards indicating better performance. The agents' learning curves were benchmarked against three baselines (Mahmood et al., 2018): The Proximal Policy Optimisation (PPO) algorithm's efficacy in our Lunar Landing simulation project is clearly shown by the sequence of graphs. The average rewards graph (Figure 2) shows a notable increasing trend at first, stabilising at higher values, suggesting effective learning and adaptability. Both measures gradually stabilise despite the variable policy loss (Figure 3) and value loss (Figure 4), which depict usual oscillations during training. This suggests that policy decisions and value predictions have been refined. The episode rewards (Figure 5) steadily increase until they reach stable, higher rewards, demonstrating the resilience of PPO in handling task complexity. When taken as a whole, these measurements validate PPO's applicability for improving performance in complex simulation settings.

1. **Optimal Performance:** An idealized scenario where an agent performs the task with maximum efficiency.
2. **Average Human Performance:** A realistic measure of how a human might be expected to perform, serving as a practical baseline.
3. **Random Agent:** A baseline where actions are selected randomly, representing the lower bound of expected performance.

For more information refer to Appendix C.

5.1 Learning Efficiency and Stability:

The performance analysis of various reinforcement learning algorithms sheds light on their efficiency in policy optimization and stability. PPO demonstrates robust and consistent performance, efficiently balancing exploration and exploitation. DQN exhibits significant variability in reward acquisition, while DDQN offers improved stability by mitigating overestimation bias. DDQN, although marginally outperforming DQN in average reward, shows less pronounced benefits from prioritized experience replay (PER). Graphical analysis underscores PPO's steady improvement trend, while DQN and DDQN display higher volatility due to value-based instability. PER's impact, although less pronounced, hints at its potential significance in optimizing learning outcomes within complex environments. For more information refer to Appendix C. The actor-critic framework of PPO demonstrates consistent learning, which qualifies it for contexts that are stable (Schulman et al., 2017). While DDQN lowers variance, it is not as stable as PPO. According to Van Hasselt, Guez and Silver (2016), prioritisation enhances performance dependent on environmental factors, highlighting the significance of algorithm selection for certain tasks. While DDQN lessens variability in conventional DQN approaches, PPO excels in the establishment of stable policies. Experience replay with a higher priority helps situations when certain experiences have a bigger educational value. For more information refer to Appendix C. These observations facilitate the efficient application of RL agents to intricate control tasks.

5.2 Comparative Analysis and Discussion

The selection of these methods was driven by their individual and collective capabilities to address the complex requirements of the "Lunar Lander" task. PPO's fast convergence and robustness make it particularly well-suited for environments where rapid and reliable learning is crucial. DQN provides a solid framework for discrete action decisions, and DDQN enhances this framework with more accurate value estimations. Additionally, the inclusion of Prioritized Experience Replay (PER) introduces an efficient sampling mechanism by prioritizing experiences based on their importance,

allowing for more effective learning from rare and high-reward experiences. This enhances the agent's ability to generalize and learn from experiences efficiently, further improving its overall performance. Each algorithm contributes uniquely to tackling the specific challenges posed by the "Lunar Lander" environment. PPO stands out due to its efficient learning curve and stability across various conditions, marking it as the preferred method for environments demanding precision and adaptability. This detailed investigation into these RL techniques not only aids in selecting appropriate algorithms for similar tasks but also enhances the broader discourse on their practical applications in complex simulations.For more information refer to 1.

Agent	Maximum score	Time to solve	Episodes to Stability
PPO	246.40	23.54	400 - 500
DQN	242.40	21.59	1000 - 1100
DDQN	261.96	17.52	800 - 900
PER	258.58	21.52	780 - 850

Table 1: "Comparison of Maximum Score, Time to Solve, and Episodes to Stability achieved by different reinforcement learning agents on the Lunar Lander task

6 Discussion

Our analysis aimed to determine the most effective RL strategy by comparing Proximal Policy Optimization (PPO), Deep Q-Network (DQN), Double Deep Q-Network (DDQN), and Prioritized Experience Replay (PER). The criteria for effectiveness included learning speed, stability of the learning process, and the overall performance in terms of landing success and fuel efficiency.

6.1 PPO's Superior Adaptation and Performance

Fast Convergence: Refer to 1 in the Appendix: PPO exhibited rapid convergence compared to DQN, PER and DDQN upon hypertuning, showcasing its efficient policy optimization capabilities. This swift adaptability is crucial in environments like "Lunar Lander," where the agent must quickly learn to navigate a complex combination of continuous state and action spaces.

Stable Learning Trajectory: The learning curves presented in the results show that PPO maintained a more consistent and stable improvement across episodes, with fewer fluctuations in performance compared to its counterparts. This stability is particularly beneficial in the "Lunar Lander" simulation, where erratic behavior can lead to catastrophic failures like crashing.

Effective Exploration-Exploitation Balance: PPO's clipping mechanism, which moderates the updates to policy gradients, proved effective in maintaining a balance between exploring new actions and exploiting known strategies. This balance is key to achieving the dual goals of landing accuracy and minimal fuel consumption.

6.2 Comparison with Other Algorithms:

While DQN and DDQN made significant improvements in their performance over time, they exhibited higher variability in rewards and were more susceptible to overestimation biases. DDQN addressed some of these issues by decoupling action selection from value estimation, yet it still lagged behind PPO in terms of the overall stability and consistency. The marginal improvements observed with PER over DQN suggest that while prioritization helps in learning from significant transitions effectively, it does not substantially overcome the fundamental challenges posed by the "Lunar Lander" task, such as managing continuous control dynamics. The ability of PPO to effectively manage the trade-offs between various control objectives (e.g., balancing speed and fuel consumption while ensuring a safe landing) aligns well with industrial applications where multiple objectives must be simultaneously optimized.

In conclusion, the application of Proximal Policy Optimization combined with an Actor-Critic framework has proven to be highly effective in solving the "Lunar Lander" problem. It outperformed other examined algorithms across several key metrics, demonstrating its suitability for complex RL tasks that require rapid and stable learning.

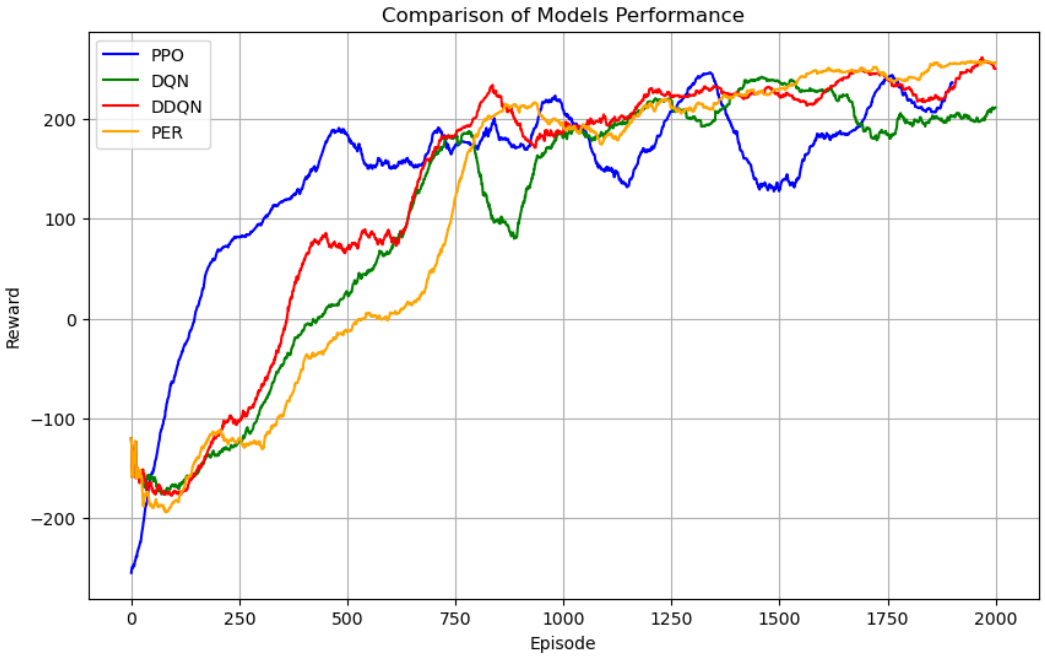

Figure 1: Comparison Graph between Different algorithms

7 Future Work

Building on the achievements of employing Proximal Policy Optimization (PPO) within an Actor-Critic framework in the "Lunar Lander" simulation, the pursuit of further enhancements is essential. Future studies might delve into combining PPO with Trust Region Policy Optimization (TRPO)(Schulman et al., 2015) or innovating with adaptive clipping mechanisms to bolster policy flexibility and adaptability. Incorporating advanced exploration methods such as curiosity-driven learning (Burda et al., 2018) and entropy maximization could enrich the exploration capabilities of the algorithms, helping to avert premature convergence and fostering more robust decision-making processes. Testing these algorithms across diverse landing scenarios and employing domain randomization (Tobin et al., 2017) may reveal deeper insights into the adaptability and resilience of these models against varying environmental conditions. Additionally, scaling up through parallel training in distributed computing environments could significantly enhance the scalability and efficiency of these algorithms, making them more viable for real-world applications. The development of comprehensive theoretical frameworks that ensure convergence and stability, coupled with experiments involving human-in-the-loop simulations(Mosqueira-Rey et al., 2023), is crucial for deepening our understanding of human-agent interactions, thereby improving the reliability and safety of autonomous systems in critical operations. These research directions not only aim to improve the performance and efficiency of reinforcement learning algorithms but also to expand their practical applications, ensuring the creation of more dependable, efficient, and safe autonomous systems.

8 Personal Experience

During our project on the Lunar Landing simulation, integrating four distinct algorithms presented significant challenges, particularly due to frequent environment errors. These included import and dependency issues, which necessitated constant troubleshooting. Each error served as a critical learning point, pushing us to delve deeper into the intricacies of our programming environment and dependencies. Rectifying these errors not only enhanced our technical skills but also underscored the importance of resilience and meticulous attention to detail in software development. The process, while often frustrating, ultimately proved to be a gratifying demonstration of our team's capability to overcome technical obstacles and achieve our objectives.

References

Brady, T. and Paschall, S., 2010. The challenge of safe lunar landing. *2010 ieee aerospace conference*. IEEE, pp.1–14.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T. and Efros, A.A., 2018. Large-scale study of curiosity-driven learning. *arxiv preprint arxiv:1808.04355*.

Fan, J., Wang, Z., Xie, Y. and Yang, Z., 2020. A theoretical analysis of deep q-learning. *Learning for dynamics and control*. PMLR, pp.486–489.

Li, J., Chen, Y., Zhao, X. and Huang, J., 2022. An improved dqn path planning algorithm. *The journal of supercomputing*, 78(1), pp.616–639.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., 2015. Continuous control with deep reinforcement learning. *arxiv preprint arxiv:1509.02971*.

Mahmood, A.R., Korenkevych, D., Vasan, G., Ma, W. and Bergstra, J., 2018. Benchmarking reinforcement learning algorithms on real-world robots. *Conference on robot learning*. PMLR, pp.561–591.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M., 2013. Playing atari with deep reinforcement learning. *arxiv preprint arxiv:1312.5602*.

Mohammad Hasani Zade, B. and Mansouri, N., 2021. Ppo: a new nature-inspired metaheuristic algorithm based on predation for optimization. *Soft computing*, pp.1–72.

Mosqueira-Rey, E., Hernández-Pereira, E., Alonso-Ríos, D., Bobes-Bascarán, J. and Fernández-Leal, Á., 2023. Human-in-the-loop machine learning: a state of the art. *Artificial intelligence review*, 56(4), pp.3005–3054.

Schaul, T., Quan, J., Antonoglou, I. and Silver, D., 2015. Prioritized experience replay. *arxiv preprint arxiv:1511.05952*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. and Moritz, P., 2015. Trust region policy optimization. *International conference on machine learning*. PMLR, pp.1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O., 2017. Proximal policy optimization algorithms. *arxiv preprint arxiv:1707.06347*.

Sutton, R.S., 1988. Learning to predict by the methods of temporal differences. *Machine learning*, 3, pp.9–44.

Sutton, R.S. and Barto, A.G., 2018. *Reinforcement learning: An introduction*. MIT press.

Sutton, R.S., McAllester, D., Singh, S. and Mansour, Y., 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Tavares, A.R. and Chaimowicz, L., 2018. Tabular reinforcement learning in real-time strategy games via options. *2018 ieee conference on computational intelligence and games (cig)*. IEEE, pp.1–8.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W. and Abbeel, P., 2017. Domain randomization for transferring deep neural networks from simulation to the real world. *2017 ieee/rsj international conference on intelligent robots and systems (iros)*. IEEE, pp.23–30.

Van Hasselt, H., Guez, A. and Silver, D., 2016. Deep reinforcement learning with double q-learning. *Proceedings of the aaai conference on artificial intelligence*. vol. 30.

Wiering, M.A. and Van Otterlo, M., 2012. Reinforcement learning. *Adaptation, learning, and optimization*, 12(3), p.729.

9 Appendices

9.1 Appendix A: Hyperparameters for DQN, DDQN, PER

Parameter Name	Value
BUFFER_SIZE	1×10^5
BATCH_SIZE	64
GAMMA	0.99
TAU	1×10^{-3}
LR	5×10^{-4}
UPDATE_EVERY	4

Hyperparameters for PPO

Parameter	Value
EPISODES	2000
GAMMA	0.99
PPO_STEPS	7
EPSILON	0.25
Learning Rate	0.001

9.2 Appendix B: Other algorithms used

9.2.1 Q-Learning

The Q-learning algorithm updates the action-value function Q using the Bellman equation as follows:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right)$$

Where:

- α is the learning rate.
- γ is the discount factor that balances the importance of immediate and future rewards.
- R_{t+1} is the reward received after taking action A_t in state S_t.
- $\max_a Q(S_{t+1}, a)$ is the maximum predicted reward attainable from the next state S_{t+1}(Sutton, 1988).

9.2.2 Actor - critic

$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right]$

The variables in the value function update formula are defined as follows:

- $V(s_t)$ is the value of being in state s at time t,
- α is the learning rate,
- r_{t+1} is the reward received after taking an action at state s_t,
- γ is the discount factor,
- s_{t+1} is the next state.

9.2.3 Soft Update Formula

Soft updates are used to gradually blend the weights of the target network θ^- with the weights of the trained network θ, ensuring smooth transitions and improving stability:

$$\theta^- = \tau\theta + (1 - \tau)\theta^-$$

Where τ is a small coefficient that determines the rate of updates to the target network (Lillicrap et al., 2015).

9.2.4 Experience Replay

Experience replay enhances learning by storing transitions (S, A, R, S') in a buffer and randomly sampling from it to break the correlation between consecutive learning updates. This method helps to utilize the experience more efficiently and learn more effectively from individual tuples multiple times.

9.3 Appendix C: Graphs for PPO

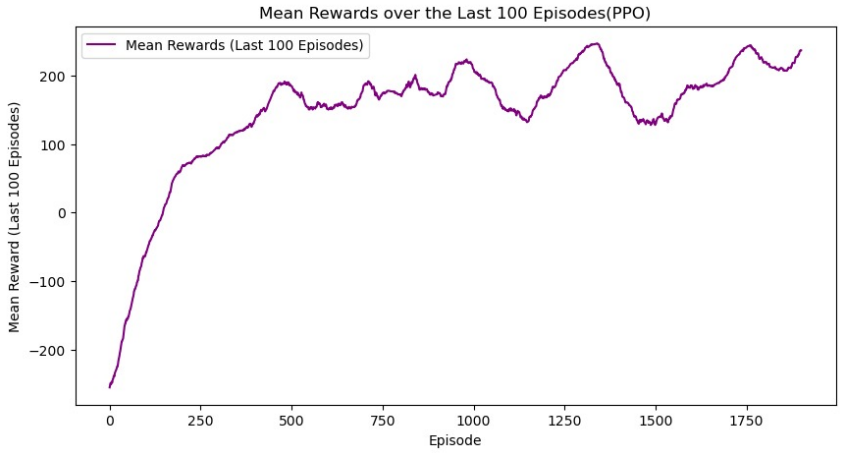

Figure 2: Mean Rewards over the last 100 Episodes for PPO

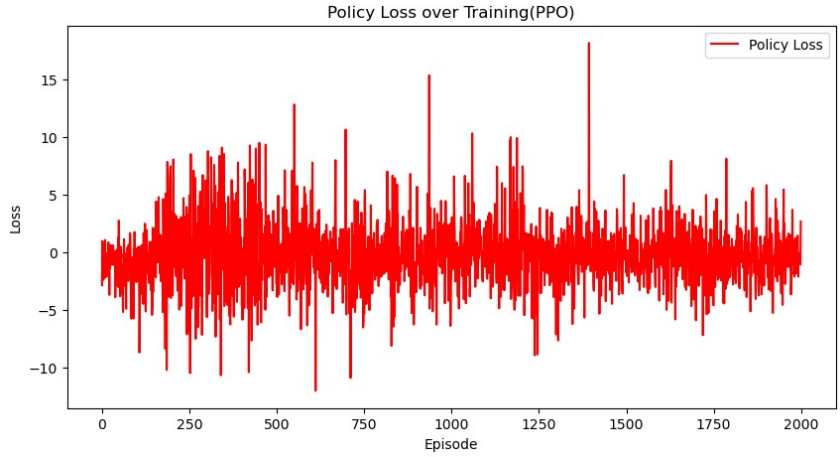

Figure 3: Policy Loss over Training PPO

9.4 Appendix D: Environments used

The experiments were conducted on an Apple M1 Pro laptop, featuring 16 GB of RAM and 512 GB of storage. The laptop's integrated 8-core CPU and 14-core GPU were utilized for computational tasks. This setup provided ample resources for conducting experiments efficiently.

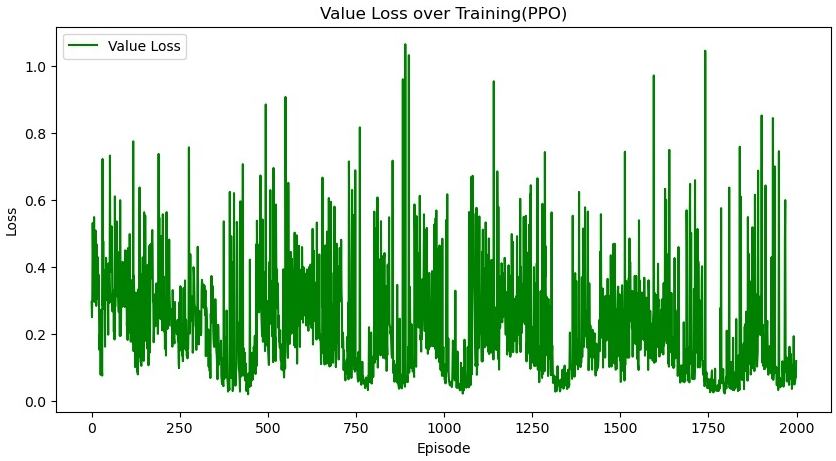

Figure 4: Value loss over Training PPO

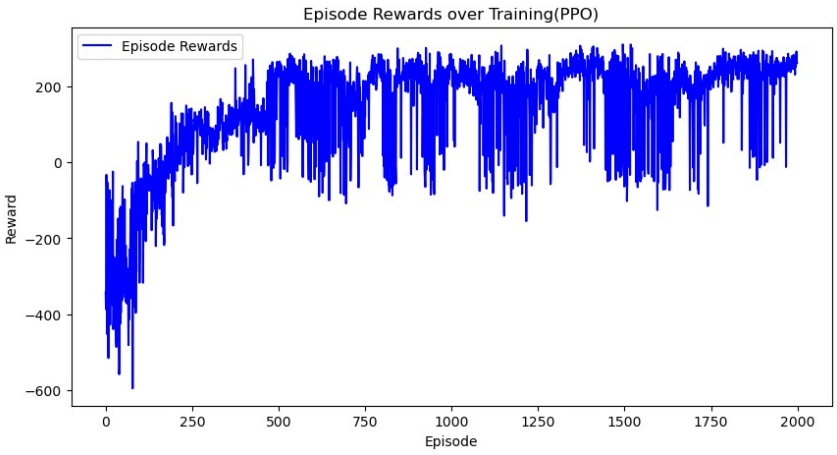

Figure 5: Episode Rewards over Training PPO