Training RL Agents in Atari Space Invaders using Stable Baselines 3

Saurav Singh Chandel[†]

[†] *Department of Mathematics and Statistics, Memorial University of Newfoundland, St. John's (NL) A1C 5S7, Canada*

E-mail: sschandel@mun.ca

Abstract: Reinforcement Learning (RL) revolutionizes artificial intelligence by providing a framework for agents to learn through interaction and rewards. Informed by the multi-armed bandit problem, this study delves into the complexities of decision-making, particularly in diverse states where the optimal action depends on the agent's current situation. Leveraging Stable Baselines, a Python module facilitating RL algorithm implementation, we employ advanced agents like Deep Q-learning (DQN) and Proximal Policy Optimization (PPO) to approximate solutions to the Bellman optimality equation. The stability and reproducibility offered by Stable Baselines allow us to concentrate on fundamental RL principles, exemplified through practical application. In this research, we train agents for the Atari Space Invaders environment, meticulously experimenting with training steps, and visualizing metrics to assess the agent's evolving performance. This study serves not only as a practical exploration into training agents for specific environments but also contributes to a broader understanding of how machines learn, adapt, and make intelligent decisions.

Keywords: Reinforcement Learning, Markov Decision Process, Bellman Optimality Equation, A2C, DQN, PPO, TD3

1 Introduction

Reinforcement Learning (RL) stands as a foundational concept in the realm of artificial intelligence, mirroring the way humans and animals learn from interactions with their environment. Unlike traditional machine learning paradigms, RL agents don't rely on a predefined dataset for training. Instead, they learn by trial and error, adapting their behavior based on the consequences of their actions. This approach makes RL particularly suitable for scenarios where decisions are made sequentially over time, such as in robotics, gaming, and autonomous systems.

At the heart of RL lies the exploration-exploitation dilemma, analogous to a decision-maker choosing between exploiting their current knowledge and exploring new possibilities. This concept finds a compelling analogy in the multi-armed bandit problem, a simplified RL scenario where an agent faces a row of slot machines, each with potentially different rewards. The challenge is to determine the most rewarding machine by experimenting with different choices and metaphorical representation of the exploration-exploitation trade-off.

However, the multi-armed bandit problem is a precursor to the full RL challenge, which introduces the notion of multiple states. In real-world scenarios, the optimal action a RL agent should take depends not only on its current state but also on the history of states and actions it has encountered. For instance, in the context of driving a car, the appropriate action to take, such as turning left, depends on the specific state of the road, whether it is curving left or if the car is at an intersection.

To address RL problems with multiple states, the formalism of Markov Decision Processes (MDPs) is often employed. MDPs provide a mathematical framework to model sequential decision-making problems, defining states, actions, and rewards. One of the key concepts within MDPs is the Bellman optimality equation, which expresses the relationship between the value of a state and the expected return from that state.

$$v^*(s) = \max_a E[R_{t+1} + v^*(S_{t+1})|S_t = s, A_t = a]$$

This equation captures the idea that the value of being in a particular state under an optimal policy is equal to the expected return of the best action from that state.

In our exploration of RL, we turn to Stable Baselines 3, a Python library built on OpenAI's Baselines. Stable Baselines 3 offers a collection of state-of-the-art RL algorithms, making it a powerful tool for training autonomous agents. The advantage of using Stable Baselines lies in its ability to handle various environments, ranging from video games and classical control tasks to more complex scenarios like self-driving cars.

In this project, we delve into training RL agents using Stable Baselines for the Atari Space Invaders environment. The objective is to apply RL algorithms, such as Deep Q-learning (DQN), Advantage Actor-Critic (A2C), or Proximal Policy Optimization (PPO), to train agents that can effectively play the Space Invaders game.

In summary, our journey through RL involves understanding its conceptual foundations, grappling with mathematical formulations like the Bellman optimality equation, and practically applying these principles using Stable Baselines. The Atari Space Invaders environment serves as a challenging testbed to observe and evaluate the learning capabilities of RL agents, showcasing the potential and versatility of Stable Baselines in training intelligent decision-making systems.

2 Methods

2.1 Reinforcement Learning Framework

The study employs a reinforcement learning framework, a paradigm within machine learning where an agent learns to make decisions by interacting with an environment. Unlike traditional supervised learning, the agent receives feedback in the form of rewards or punishments, guiding it toward optimal decision-making strategies.

2.2 Markov Decision Processes (MDP)

To model the sequential decision-making process, we adopt the framework of finite Markov decision processes (MDP). In MDP, the agent interacts with the environment at discrete time steps, receiving a representation of the environment's state and selecting an action based on this information. The agent then transitions to the next time step, receiving a numerical reward and arriving in a new state.

2.3 Bellman Optimality Equation

The central theoretical foundation of our approach is the Bellman optimality equation. This equation expresses that the value of a state under an optimal policy must be equal to the expected return for the best action from that state. Mathematically, for the value function of a state s under the policy π, denoted as $v_\pi(s)$, it is defined as

$$v_\pi(s) = E\left(\sum_{k=t+1}^{T} R_k \middle| S_t = s\right)$$

Similarly, the action-value of a state-action pair (s, a) under the policy π is given by:

$$q_\pi(s, a) = E\left(\sum_{k=t+1}^{T} R_k \middle| S_t = s, A_t = a\right)$$

2.4 Stable Baselines and Gym

To implement and experiment with reinforcement learning algorithms, we leverage the Stable Baselines library, version 3 [2]. Stable Baselines is a Python module, a fork of the OpenAI baselines module, designed to facilitate the implementation of various reinforcement learning algorithms in PyTorch. The library provides a collection of modern reinforcement learning agents, including Deep Q-learning (DQN), Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C) and Twin Delayed DDPG (TD3).

2.4.1 Deep Q-learning (DQN)

- **Objective Function:** The objective is to learn a Q-function, denoted as $Q(s, a)$, which estimates the expected cumulative future rewards for taking action a in state s.

- **Loss Function:** The loss function is often defined as the temporal difference (TD) error, represented as $L(\theta) = \mathbb{E}[(r+\gamma \max_{a'} Q(s', a'; \theta^-)-Q(s, a; \theta))^2]$, where θ are the parameters of the Q-network, r is the immediate reward, s' is the next state, and γ is the discount factor. [4]

2.4.2 Proximal Policy Optimization (PPO)

- **Objective Function:** PPO aims to maximize the expected cumulative rewards by optimizing a policy parameterized by θ.
- **Surrogate Objective:** The objective is to maximize the surrogate function, defined as $L^{CLIP}(\theta) = \mathbb{E}[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$, where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{old}}(a_t|s_t)}$ is the likelihood ratio, and $\hat{A}_t$ is the advantage function. [5]

2.4.3 Advantage Actor-Critic (A2C)

- **Objective Function:** A2C combines policy optimization (actor) and value estimation (critic) to maximize expected cumulative rewards.
- **Actor Loss:** The actor loss is typically the policy gradient with an advantage term: $L^{actor}(\theta) = -\mathbb{E}[\log(\pi_\theta(a_t|s_t))\hat{A}_t]$, where $\pi_\theta(a_t|s_t)$ is the policy and $\hat{A}_t$ is the advantage.
- **Critic Loss:** The critic loss is the mean squared error (MSE) loss for the value estimation: $L^{critic}(\theta) = \frac{1}{2}\mathbb{E}[(V_\theta(s_t) - V_t)^2]$, where $V_\theta(s_t)$ is the predicted value and V_t is the estimated value. [6]

2.4.4 Twin Delayed DDPG (TD3)

- **Objective Function:** TD3 is an extension of DDPG, aiming to optimize a deterministic policy μ_θ and a value function Q_ϕ.
- **Q-function Loss:** The Q-function loss is defined as $L_Q(\phi) = \mathbb{E}[(Q_\phi(s_t, a_t) - y)^2]$, where $y = r_t + \gamma(1 - d_t)Q_{\phi'}(s_{t+1}, \mu_{\theta'}(s_{t+1}))$ is the TD target, d_t is a binary discount term, and ϕ' and θ' are target network parameters.
- **Policy Loss:** The policy loss encourages the policy to maximize the Q-function: $L_\pi(\theta) = -\mathbb{E}[Q_\phi(s_t, \mu_\theta(s_t))]$. [3]

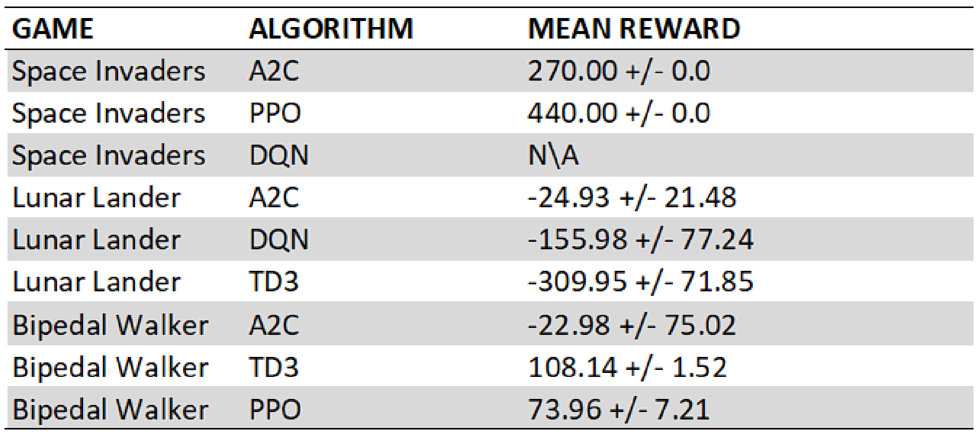
In addition to Stable Baselines, we utilize the Gym toolkit2, an open-source toolkit for developing and comparing reinforcement learning algorithms. Gym provides a diverse set of environments, allowing us to train agents in scenarios ranging from classic control tasks to complex video game environments, such as the Atari Space Invaders game. [1]

3 Results

3.1 Training Agents

3.1.1 Atari Space Invaders

In our study of training computers to play Atari Space Invaders, we tested two smart methods: Proximal Policy Optimization (PPO) and Advantage Actor-Critic (A2C). PPO seemed effective in improving our computer player's skills, and A2C showed promise in learning too. However, when it came to another method called Deep Q-learning (DQN), we faced challenges. DQN required a lot of memory, and our computer struggled to handle the game's complexity. The results for A2C and PPO were a bit confusing and didn't make much sense, indicating that more work is needed to understand or fix the outcomes. Despite these challenges, the experience highlighted the critical importance of managing computer memory, especially when teaching them intricate games like Atari Space Invaders.



GAME	ALGORITHM	MEAN REWARD
Space Invaders	A2C	270.00 +/- 0.0
Space Invaders	PPO	440.00 +/- 0.0
Space Invaders	DQN	N\A
Lunar Lander	A2C	-24.93 +/- 21.48
Lunar Lander	DQN	-155.98 +/- 77.24
Lunar Lander	TD3	-309.95 +/- 71.85
Bipedal Walker	A2C	-22.98 +/- 75.02
Bipedal Walker	TD3	108.14 +/- 1.52
Bipedal Walker	PPO	73.96 +/- 7.21

Figure 1. Mean Rewards

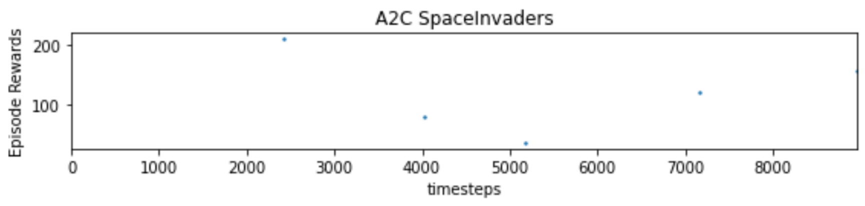


Figure 2. Mean Rewards (A2C)

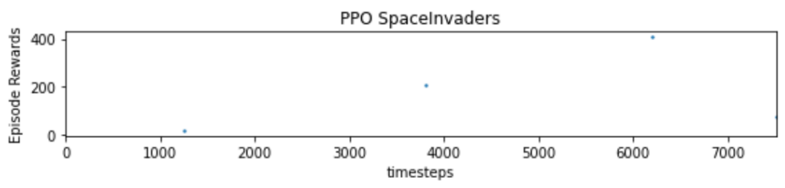


Figure 3. Mean Rewards (PPO)

3.1.2 Lunar Lander

In our study to teach computers to land on the Moon in a game called LunarLander, we tried three different ways: Twin Delayed DDPG (TD3), Deep Q-learning (DQN), and Advantage Actor-Critic (A2C). TD3 was pretty good at helping the computer learn, especially in dealing with tricky actions. However, when we looked at the graphs that showed how well the computer was doing, they were a bit confusing. Right now, I'm not sure what they mean, so we need to do more research to figure it out. A2C also showed it could learn, but DQN had a bit of a hard time with the LunarLander game. So, we see that choosing the right way for the computer to learn is important, and we're still figuring it out.

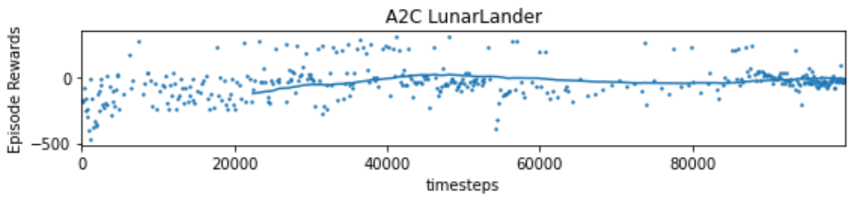


Figure 4. Mean Rewards (A2C)

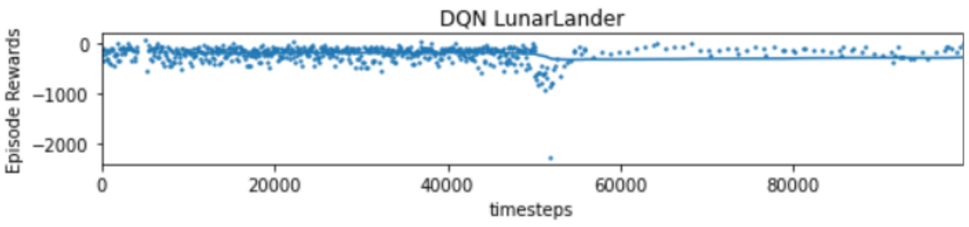


Figure 5. Mean Rewards (DQN)

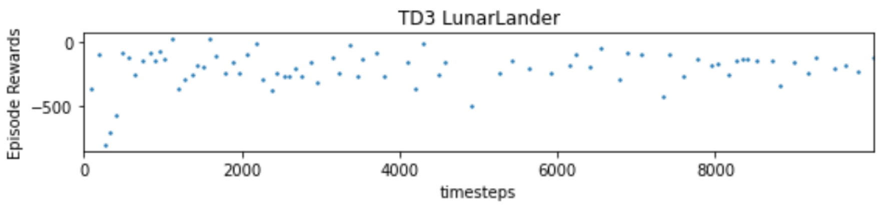


Figure 6. Mean Rewards (TD3)

3.1.3 Bipedal Walker

In our exploration of training computers to navigate the Bipedal Walker game, we employed three different learning methods: Advantage Actor-Critic (A2C), Twin Delayed DDPG (TD3), and Proximal Policy Optimization (PPO). A2C demonstrated promise in enhancing the walking abilities of our computer-controlled agent, while TD3 and PPO exhibited ambiguous results. However, the interpretability of the outcomes proved challenging due to complexity or potential issues in plotting. The ambitious results suggest that additional training steps or improved plotting techniques may be necessary for a clearer understanding of the agent's performance. This underscores the need for ongoing refinement and investigation to optimize the training process for computer-controlled bipedal navigation.

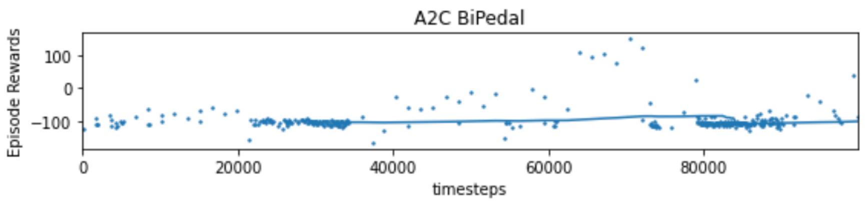


Figure 7. Mean Rewards (A2C)

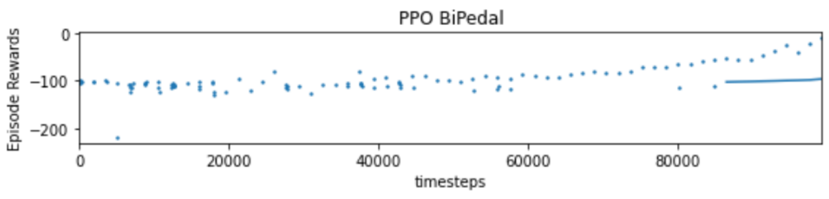


Figure 8. Mean Rewards (PPO)

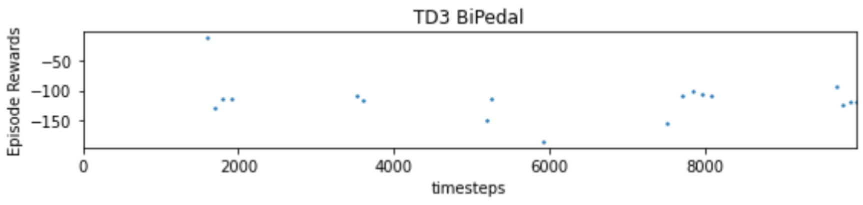


Figure 9. Mean Rewards (TD3)

4 Conclusion

In conclusion, our study on training computers across Atari Space Invaders, LunarLander, and Bipedal Walker games has yielded varied outcomes. Proximal Policy Optimization (PPO) and Advantage Actor-Critic (A2C) demonstrated promise in improving computer player skills in Atari Space Invaders, although further investigation is required to make sense of the results. LunarLander training with Twin Delayed DDPG (TD3) showcased effectiveness, but the graphical representations proved challenging to interpret, necessitating additional research for clearer insights. Bipedal Walker training using A2C exhibited promising walking abilities, while TD3 and PPO yielded ambiguous results, suggesting the need for more training steps or enhanced plotting techniques. In all cases, these findings highlight the complexity of training computer agents in diverse environments, emphasizing the ongoing need for refinement and deeper exploration to attain more concrete and comprehensible results.

Acknowledgements

I want to sincerely thank Professor Alexandar Bihlo for his excellent mentoring and instruction, which gave me a solid foundation in mathematics and reinforcement learning. Additionally, I thank the creators of Stable Baselines for developing a powerful and user-friendly framework that greatly facilitated the implementation of reinforcement learning algorithms. The success of this project was greatly aided by the abundance of materials offered by the Stable Baselines documentation. Lastly, I appreciate the assistance and insights provided by ChatGPT, which proved to be a valuable resource in addressing queries and refining ideas throughout the project. [7]

References

[1] Gym documentation, https://gym.openai.com/docs/.

[2] Stable baselines documentation, https://stable-baselines3.readthedocs.io/en/master/.

[3] AI O., Td3 (twin delayed ddpg) - spinning up documentation, https://spinningup.openai.com/en/latest/algorithms/td3.html.

[4] Farebrother J., Machado M.C. and Bowling M., Generalization and regularization in dqn, *arXiv preprint arXiv:1810.00123* (2018).

[5] Han S.Y. and Liang T., Reinforcement-learning-based vibration control for a vehicle semi-active suspension system via the ppo approach, *Applied Sciences* **12** (2022), 3078.

[6] Li X., Chen G., Wu G., Sun Z. and Chen G., Research on multi-agent d2d communication resource allocation algorithm based on a2c, *Electronics* **12** (2023), 360.

[7] OpenAI, Gpt-3, a powerful language model, https://www.openai.com, 2022.

Appendix

Link to the demonstration video