Developing an Intelligent Flappy Bird Player Using Reinforcement Learning Algorithms

Bhargavi Poyekar, Arya Honraopatil, Prerana Bollineni, Mukesh Kumar Vidam
University of Maryland, Baltimore County
bpoyeka1@umbc.edu, ahonrao1@umbc.edu, preranb1@umbc.edu, mukeshv1@umbc.edu

Abstract

Flappy Bird is a simple, yet challenging game. It presents a complex problem for artificial intelligence (AI) agents seeking to navigate a bird through a maze of pipes. In this study, we tried to build an intelligent Flappy Bird player through the application and analysis of various reinforcement learning (RL) algorithms. The objective was to explore the efficacy of these algorithms based on performance measures such as average reward obtained over episodes, learning curve, and training time in training an AI agent to successfully pass through the dynamically changing environment of the Flappy Bird. Throughout the study, we implemented and evaluated multiple RL approaches, encompassing Q-learning, Deep Q-Networks (DQN), and Policy Gradient. Notably, our findings reveal that the DQN algorithm emerged as the most promising, demonstrating superior performance. It achieved a maximum reward of 14 over the minimum number of iterations compared to the other two algorithms. This study's outcomes provide valuable insights into the strengths and limitations of different RL techniques.

Introduction

Flappy Bird requires players to guide a bird through a series of pipes, maintaining flight without hitting the pipes or the ground. Supervised learning cannot be used here as there are very few positive examples to learn from. Using Reinforcement learning, the agent can learn to avoid the pipes by getting rewards when it successfully passes through the pipes and a penalty is charged when it collides. The agent has to process the information about the bird's position and position of the pipes and various other environmental factors in real time. The relation between the player and the outcome is highly nonlinear and complex. Hence, a trial-and-error approach of reinforcement learning would work better in this case.

Flappy Bird is also a sequential decision-making problem where the agent needs to learn a strategy over time. Reinforcement learning algorithms can optimize a policy, which is a strategy that maps states to actions through the interactions with its environment. This is crucial in learning the optimal strategy for navigating the bird through the pipes. We have developed an intelligent Flappy Bird agent using the following reinforcement learning algorithms: Q-learning Watkins and Dayan (1992), Deep Q Network (DQN) Mnih et al. (2013), and Policy-gradient Williams (1992) approach. We have compared the efficiency of these algorithms based on performance measures such as average reward obtained over episodes, learning curve, and training time. The user interface includes a visually appealing representation of the Flappy Bird game.

Related Work

Reinforcement learning has a prominent focus within the expansive field of artificial intelligence research. In this, an agent explores potential strategies or actions within a designated environment, gathering feedback in the form of rewards or costs based on the outcomes of its decisions. Subsequently, the agent derives a behavioral policy from these observations. Deep Reinforcement Learning (DRL) combines traditional reinforcement learning with the capabilities of deep learning methodologies. Recent advancements in DRL have yielded more adept intelligent agents which are capable of tackling challenges involving high-dimensional input data, such as images, with reasonable computational resources as shown by Mnih et al. (2015).

Active Neural Generative Coding (ANGC) framework introduced by Ororbia and Mali (2022) is built for learning of goal-directed agents without relying on backpropagation of errors. It is effective on various control problems and shows better performance with deep Q-learning methods. The authors prove that ANGC operates successfully even in environments with sparse rewards and draws inspiration from the cognitive theory of planning as inference.

Neuroevolution was used by Williams (1992) in combination with reinforcement learning to get a high score. Using distance and velocity as inputs, as opposed to high-dimensional image data, contributed to their high scores. Convergence analysis for algorithms using discounted advantage estimations (DAE) and a fictitious discount factor in the score functions of policy gradient estimators enhances the understanding of the nuances of the DRL algorithms-Guo, Hu, and Zhang (2022). The authors establish quantitative connections between the three settings of Markov decision processes (MDPs): finite-time-horizon, average reward, and discounted settings.

Experiments with different optimizers have been done to

improve the performance of DRL algorithms as done by He (2022). It examines how Adam and RMSProp optimizers work during training and concludes that although Adam optimizer performs better at first with fewer training rounds, RMSProp performs noticeably better with more training rounds. In the context of training in a simulation environment, Mohamed, Mohamed, and Mohamed (2021) present the application of Proximity Policy Optimization (PPO) on a simulated quadrotor navigated through Unity and Microsoft Airsim. They focus on designing payment systems and understanding discount trends based on the environment.

In this project, we investigate and elaborate on several of these hypotheses to build an intelligent agent to provide an overview on the strengths and limitations of different DRL techniques.

Data

We obtained the Flappy Bird Game from the GitHub repository [1] as shown in Figure 1. The primary data required for the project was the game state information, including the bird's position, the position of pipes, and relevant environmental variables. For the DQN algorithm, we modified the code to extract the state information as an image at each unit of time (seconds). To remove unnecessary visual features and isolate the important gaming area, the raw picture is cropped. The picture is then scaled to 84X84 pixels, a uniform square dimension that guarantees input size constancy for efficient processing. By reducing the image to a single channel, grayscale conversion preserves important visual details. By assigning non-zero pixel values to 255, binarization further streamlines the picture and highlights important elements of the learning process. After processing, the picture is transformed into a PyTorch tensor by converting it to a float32 data type and transposing its dimensions to match PyTorch's required format (channels, height, and width). The resulting tensor represents a brief but informative moment in time when the game state was at that particular point. The processed pictures are concatenated to form a sequence, which becomes the input representation for the neural network and helps in learning across successive time steps. We utilized SIFT to extract the bird and pipe locations' coordinates, which we then provided as a state input to the Q-learning and Policy-gradient algorithms. During the training phase, we created synthetic data by interacting with the game environment, even though there was no pre-existing dataset specifically for Flappy Bird. This approach allowed the agent to learn and adapt to the unique dynamics of Flappy Bird, ensuring its ability to generalize across various game scenarios.

Methods

Our approach to reinforcement learning algorithms involved a strategic selection that encompasses diverse methodologies. The Policy Gradient Williams (1992) algorithm employs gradient ascent to optimize the agent's policy, while

[1]Marqs, L. 2022. Flappy Bird Python. GitHub. `https://github.com/LeonMarqs/Flappy-bird-python`. Accessed: Nov 20, 2023.

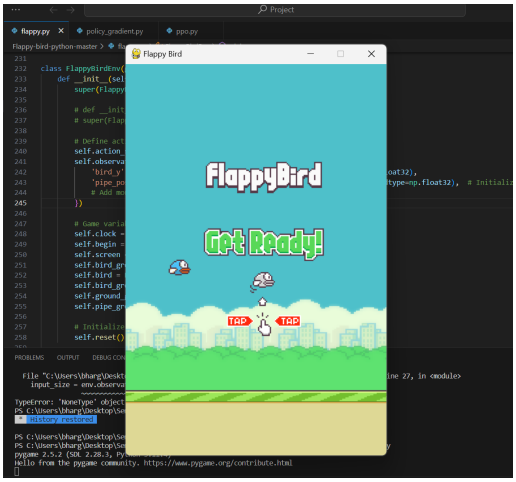

Figure 1: Flappy Bird Game

the Proximal Policy Optimization (PPO) Schulman et al. (2017) algorithm strikes a balance between exploration of new policies and exploitation of the current best policy. Trust Region Policy Optimization (TRPO) Schulman et al. (2015) ensures that policy updates remain within a trust region, preserving stability. Deep Q Network (DQN) Mnih et al. (2013) combines neural networks with Q-learning for value-based decision-making, and Categorical DQN Bellemare, Dabney, and Munos (2017) (C51) extends DQN to handle probability distributions over possible rewards. The Twin Delayed DDPG (TD3) Fujimoto, Hoof, and Meger (2018) algorithm enhances Deep Deterministic Policy Gradients with a twin critical structure. But, due to time constraints, we strategically narrowed down our reinforcement learning algorithm selection to three key methodologies: Policy Gradient, Q-learning, and Deep Q Network (DQN), shown in Figure 2.

In the initial project phases, we focused on establishing the reinforcement learning framework, particularly immersing ourselves in the Deep Q-Network (DQN) method Mnih et al. (2013).

The DQN training as shown in Algorithm 1 involved a robust neural network capturing relationships between states, actions, and rewards. States, reflecting critical elements like the bird's position and pipe locations, formed the foundation for the agent's perception. The action space, consisting of the choices to flap or not flap, is intricately linked to the agent's decision-making process. The Neural Network architecture comprises convolutional and fully connected layers. The input layer, accepting a stack of four consecutive frames, allows the network to capture temporal dependencies. Convolutional layers extract hierarchical features from the input frames, while Rectified Linear Unit (ReLU) activation layers introduce non-linearity. Fully connected layers process the flattened output, producing the final decision-making output with two units representing the available actions in the game. Trained with parameters such as gamma for discounting future rewards and an exploration-exploitation trade-off (epsilon), the network learned to associate state information. In the Flappy Bird environment,

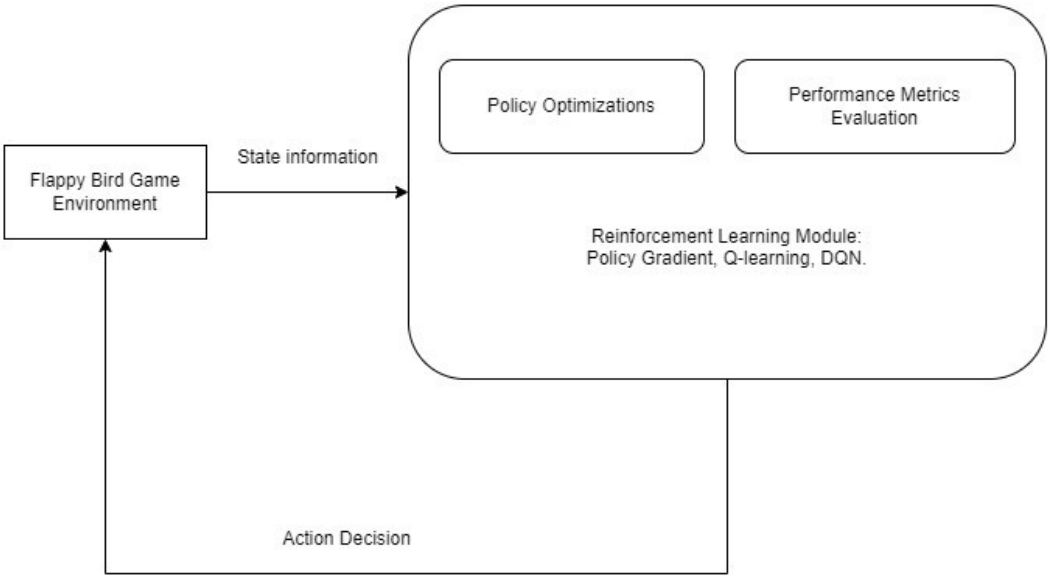

Figure 2: System Diagram of our Intelligent Flappy Bird

Algorithm 1 DQN Training

Initialize M, replay memory, and hyper parameters
Initialize ϵ, iteration, Adam optimizer, and loss criterion
Set initial action, obtain initial state s
while iteration $<$ number_of_iterations **do**
    Obtain Q from M for the current state s
    Apply ϵ-greedy exploration to select action a
    Execute a, observe s' and r
    Store transition $(s, a, r, s', \text{terminal})$ in replay memory
    if replay memory is full **then**
        Remove oldest transition from replay memory
    end if
    Anneal ϵ from $\epsilon_{\text{initial}}$ to ϵ_{final}
    Sample minibatch, compute target values y_j, and update M weights
    Set $s \leftarrow s'$, increment iteration
    Save current model and collect training info
end while

the convolutional layers enabled the recognition of spatial relationships and critical features, allowing the agent to navigate obstacles effectively. Fine-tuning parameters, such as discount factors and rewards, during training, enhanced the network's adaptability to the sparse reward structure of Flappy Bird. Through iterative learning, the neural network refines its understanding of the game dynamics, improving the agent's decision-making and overall performance in playing Flappy Bird. The algorithm employed systematic trial and error to iteratively learn optimal policies.

We opted to train the algorithm with a high Frames Per Second (FPS) setting of 99999 to expedite the training process. This decision was driven by the desire to accelerate training, considering the typically time-consuming nature of Reinforcement Learning algorithms.

After completing DQN training for our Flappy Bird agent, and exposing it to dynamic complexities for iterative learning, we transitioned to implement policy-gradient and Q-learning algorithms. This approach, outlined in Figure 3, emphasizes continual refinement and adaptation for optimal performance in the Flappy Bird environment. Q-learning is a model-free reinforcement learning algorithm that focuses on learning the optimal action-value function $Q(s, a)$, where s is the state and a is the action. The Q-learning update rule Watkins and Dayan (1992) is given by:

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

Here, r is the immediate reward, s' is the next state, α is the learning rate, and γ is the discount factor.

The key difference between Q-learning and DQN lies in the representation of the Q-function. Q-learning typically works with tabular representations for state-action pairs,

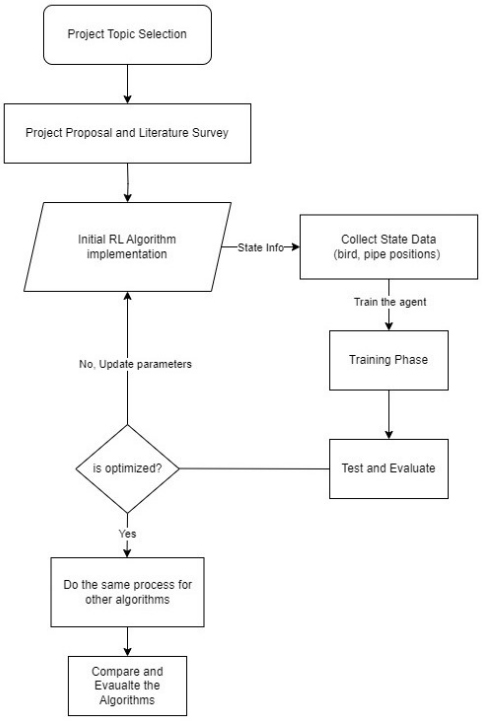

Figure 3: Flow Diagram of our Flappy Bird Project

making it well-suited for environments with discrete state and action spaces. In contrast, DQN employs a neural network to approximate the Q-function, enabling it to handle continuous state spaces.

Policy Gradient methods directly parameterize the policy and optimize it to maximize the expected cumulative reward. The objective function for policy gradient Williams (1992) is given by:

$$J(\theta) = E\left[\sum_{t=0}^{T} \nabla_\theta \log(\pi_\theta(a_t|s_t))G_t\right] \tag{2}$$

where θ represents the policy parameters, $\pi_\theta(a_t|s_t)$ is the probability of taking action a_t in state s_t, and G_t is the return.

Policy Gradient algorithms, unlike Q-learning, directly optimize the policy without explicitly estimating the action-value function. This approach makes policy gradient methods well-suited for problems with high-dimensional action spaces and stochastic policies.

The choice between Q-learning, Policy Gradient, and DQN depends on the characteristics of the environment and the nature of the problem at hand. For Flappy Bird, which involves discrete actions, Q-learning and Policy Gradient methods provide alternative strategies for training the agent effectively.

Evaluation

Performance metrics are playing a crucial role in evaluating the efficacy of our approach. For successful navigation through the gap between the upper and lower pipes, the agent receives a positive reward of 1. Additionally, if the bird aligns its horizontal position precisely with the center of a pair of pipes, an extra reward of 2 is granted, incentivizing the agent to consistently aim for the optimal path. Conversely, collisions result in unfavorable outcomes for the agent. A penalty of -1 is assigned when the bird collides with either the upper or lower pipes or reaches the ground (base), triggering a terminal state. We are measuring the agent's success through the 'average reward' metric. The 'average reward' metric serves as a comprehensive measure of the agent's success over episodes, offering insights into how well it is adapting to the sporadic and dynamic nature of rewards in the game. The 'learning curve' analysis allows us to track the progression of the agent's performance throughout training, providing insights into the learning trajectory, adaptability, and potential challenges like over-fitting or under-fitting. Additionally, we are evaluating the efficiency of each algorithm based on the 'training time' metric, which is imperative to understand the computational efficiency of each algorithm, ensuring practical and feasible implementation of the reinforcement learning model within resource constraints.

Results

Figure 4 illustrates the learning curves of the three algorithms, depicting the rewards acquired in each iteration. The training spanned a total of 850,000 iterations, with rewards displayed for every 10,000th iteration. For all three algorithms, in the initial 200,000 iterations, the agent grappled with collisions at the first pipe but gradually learned to navigate it. Subsequently, from 200,000 to 600,000 iterations, the agents focused on overcoming challenges posed by the 2nd and 3rd pipes. With continued training over 13.17 hours, the DQN agent demonstrated proficiency in passing up to pipes 8-9 without collisions. Whereas the Policy-gradient and Q-learning agents took over 15.5 and 16.8 hours respectively and still couldn't pass the 6th pipe without collision. They gained a maximum reward of 10 and 8.5 respectively.

In Figure 5, the average rewards gained over five-minute intervals for the DQN algorithm are presented. This plot further highlights the initial challenges faced by the agent, followed by a notable improvement in reward accumulation as the training progressed.

Discussion

During the project, we initially considered utilizing the Open AI Gym Brockman et al. (2016) library for developing and comparing reinforcement learning algorithms. However, compatibility issues with required libraries prompted us to forego its use. Additionally, although we initially planned to implement a range of reinforcement learning algorithms as mentioned in Methods section, time constraints led to scaling down our focus primarily on DQN, Q-learning, and policy-gradient.

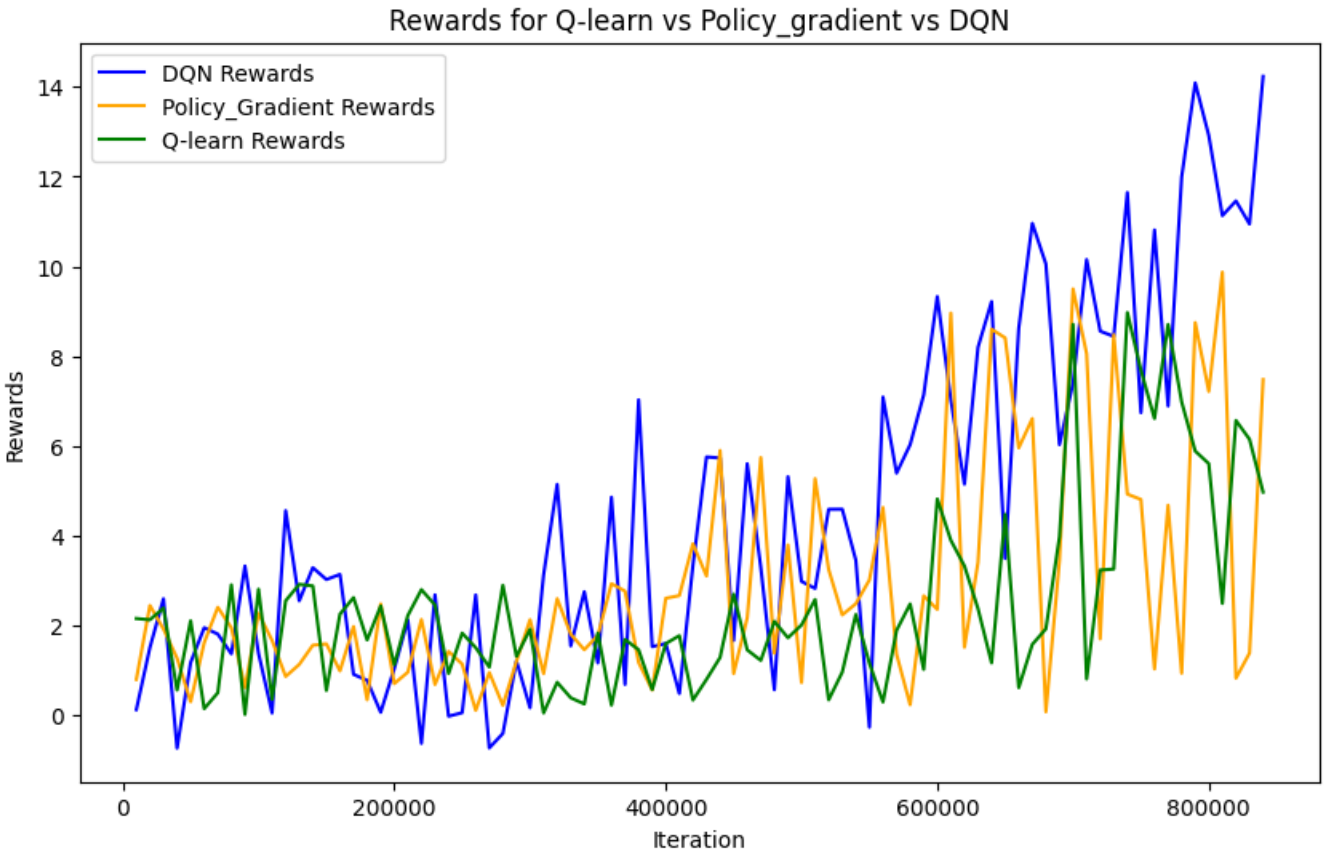

Figure 4: Learning Curve of the Reinforcement learning Algorithms

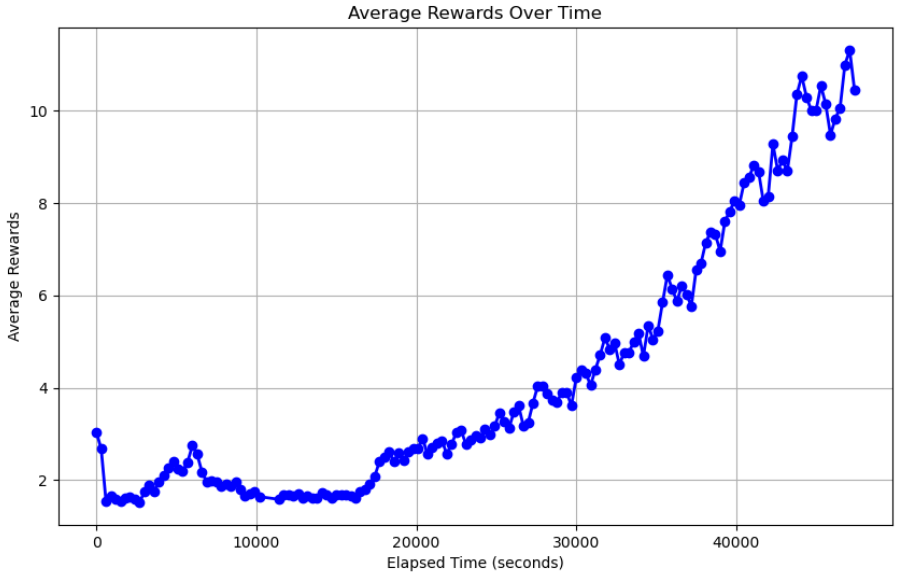

Figure 5: Average Reward of DQN Algorithm

If starting over from scratch, a more thorough evaluation of algorithmic choices and their compatibility with project requirements could streamline the selection process from the outset. One notable adjustment made during the project was the decision to increase the Frames Per Second (FPS) setting from 30 to 99999. This modification proved crucial in accelerating the training process significantly. In hindsight, if we had opted for the higher FPS setting from the project's inception, it would have expedited the training process right from the start. This realization underscores the importance of considering and optimizing such parameters early on in the project timeline to enhance overall training efficiency.

For those continuing this project, promising next steps and future work include completing the implementation of other reinforcement algorithms like TD3 and C51 to conduct a comprehensive comparative analysis with our algorithms. Further fine-tuning hyper parameters for each algorithm to enhance their adaptability and learning efficiency in the Flappy Bird environment is recommended. Exploring transfer learning, ensemble methods, and the potential of parallelizing training across multiple instances could contribute to more efficient and effective reinforcement learning agents. Additionally, extending the evaluation to other game environments within the Open AI Gym or similar frameworks could provide insights into the generalizability of the trained models.

Acknowledgments

We would like to thank Dr. Lara Martin for her support throughout the project.

References

Bellemare, M. G.; Dabney, W.; and Munos, R. 2017. A distributional perspective on reinforcement learning. In *International conference on machine learning*, 449–458. PMLR.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

Guo, X.; Hu, A.; and Zhang, J. 2022. Theoretical guarantees of fictitious discount algorithms for episodic reinforcement learning and global convergence of policy gradient methods. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 6774–6782.

He, Z. 2022. Analysis on deep reinforcement learning with flappy brid gameplay. In *2022 5th International Conference on Information and Computer Technologies (ICICT)*, 95–99. IEEE.

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*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. volume 518, 529–533. Nature Publishing Group.

Mohamed, F. A.; Mohamed, S. W.; and Mohamed, A. M. 2021. Learning to fly with deep reinforcement learning. In *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, 1–7. IEEE.

Ororbia, A. G., and Mali, A. 2022. Backprop-free reinforcement learning with active neural generative coding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 29–37.

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

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

Watkins, C. J. C. H., and Dayan, P. 1992. Q-learning. *Machine Learning*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8:229–256.