Capstone Report

Judith Barrios-Rodriguez, Vanessa Bersola, Timothy Williams, Omied Willis  
Houston Community College

ITAI 1378 Computer Vision

Prof. Patricia McManus

August 08, 2024

Capstone Project: Train an AI Agent to Play Flappy Bird

# Environment Setup

In the arcade-style game Flappy Bird, the player controls Faby, the bird that keeps moving to the right. It is up to the player to guide Faby between pairs of pipes with uniformly sized gaps positioned at odd heights. Faby lowers automatically and only rises in response to a touchscreen tap from the player. The player receives one point for each pair of pipes they successfully pass through. The game finishes when the player collides with a pipe or the ground [1].

First, we created a Deep Q-Network with PyTorch in Fig 1 below. Here we defined our layer sizes, as well as a forwarding function that defined how our inputs pass through network to produce our eventual Q-values [2]. Separately in our agent.py file, we imported ‘gymnasium’, which itself is a fork of Open AI’s Gym: a toolkit for developing and comparing reinforcement learning algorithms. It provides a standardized interface for environments and can be used to structure the interaction between the AI agent and the game.

A screenshot of a computer program

Description automatically generated

Fig 1 - DQN definition

# Model Usage

The Deep Q-Network (DQN) is used to approximate the optimal action-value function, which tells the agent what action to take under certain circumstances to maximize the cumulative future reward. In the context of Flappy Bird, our model learns to decide when to flap the bird upwards to navigate between the pipes. The input layer represents the current state of the game, such as the bird's position and the distances to the next pipes. Then, the hidden layers capture the complex relationships and dynamics of the game environment, using ReLU activation for non-linear transformations. Finally, the output layer produces Q-values for possible actions (e.g., flap or no flap), guiding the agent's decisions.

# Reinforcement Learning Implementation

Our reinforcement learning implementation leveraged these primary constructs:

* **Experience Replay**: Defined in experience\_replay.py below [3], this mechanism allows us to store and replay past experiences to improve the learning process. It helps in stabilizing the learning by reusing past data.

A computer screen shot of a program code

Description automatically generated

Fig 2 - experience\_reply.py

* **Epsilon-Greedy Policy**: A strategy for balancing exploration and exploitation. The agent randomly explores with probability ε and exploits the best-known action with probability 1-ε.
* **Target Network**: A separate network used to calculate the target Q-values, updated less frequently to maintain stability during training.

# Model Training

The model will be looped many times, preferably a high number like 1000 instances or until the model is manually stopped. The pipes are generated randomly to avoid the model simply memorizing the pattern. We would implement a reward system for every time the bird stays alive (+0.1), the bird successfully passing a pipe (+1.0) and points deducted every time the bird dies (-1.0) or touches the top of the screen (-0.5). Dying would imply that the bird has touched a bounding box of a pipe or flies too high in which the model ends, and the loop starts over a new run. Each episode is calculated, and the highest reward score is the one with the longest run.

# Testing and Evaluation

Once the model has completed several runs, we would compare the worst one with the least number of rewards with the best run. If we don’t want the model to continue indefinitely, we can set a maximum reward of 10000 which would automatically stop the model. Let's say for example, the worst run was only 3 pipes crossed and the best was 19. Since the pipes are generated randomly, we can assume in the worst run that it hasn’t come across a particular configuration very often, and thus has not learned how to avoid the pipe yet. The more the model has been trained, it can take less time calculating what the best step to take after each pass of the pipe.

# Detailed account of challenges faced, and solutions attempted/implemented

Throughout the process of attempting to train an AI agent to play flappy bird  
 our team faced challenges in trying to code and implement the best practices for the results to come out accurately. The primary challenge was designing an effective reinforcement learning model that can handle the game’s stochastic environment and rapid decision-making requirements. A solution we came for this is to use deep Q-learning, which is where AI learns optimal actions by receiving prizes when it plays well, or punishments for failures. Another challenge was managing the high-dimensional input space, as the AI needs to process pixel data from the game. This can be solved by using pre-processing techniques, such as resizing images and grayscale conversion can help the input, and convolutional neural networks can extract relevant features. Additionally, training stability was improved by adding techniques like experience replay and target networks, these approaches enable AI to learn and adapt its capabilities to effectively master Flappy Bird.

Reflective entries on key learnings and insights gained throughout the project

The Flappy Bird project provided several key learnings and insights emerged. We gained a deeper understanding of reinforcement learning principles, particularly how an agent can learn optimal strategies through trial and error by balancing exploration and exploitation. Implementing deep Q-learning demonstrated the importance of feature extraction and the role of conventional neural networks in processing complex inputs. The use of gymnasium in OpenAI’s Gym gave a wide range of environments for testing and developing reinforcement learning algorithms. This allowed users to easily reset, act, and observe outcomes, this facilitates robust development and evaluation of AI models. The project provided the significance of training stability techniques like target networks in achieving consistent performance improvements. Overall, the experience with real-time feedback and model tuning gave us valuable insights of both theoretical and practical aspects of machine learning and AI advancements.

# References

Wikipedia Contributors. (2019, February 25). *Flappy Bird*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Flappy\_Bird

DatOmied. (2024). *dqn\_pytorch/dqn.py at main · DatOmied/dqn\_pytorch*. GitHub. https://github.com/DatOmied/dqn\_pytorch/blob/main/dqn.py

# Capstone Learning Log

Judith Barrios-Rodriguez, Vanessa Bersola, Timothy Williams, Omied Willis  
Houston Community College

ITAI 1378 Computer Vision

Prof. Patricia McManus

August 08, 2024

## Learning Log

Throughout the process of attempting to train an AI agent to play flappy bird  
our team faced challenges in trying to code and implement the best practices for the results to come out accurately. The primary challenge was designing an effective reinforcement learning model that can handle the game’s stochastic environment and rapid decision-making requirements. A solution we came for this is to use deep Q-learning, which is where AI learns optimal actions by receiving prizes when it plays well, or punishments for failures. Another challenge was managing the high-dimensional input space, as the AI needs to process pixel data from the game. This can be solved by   using pre-processing techniques, such as resizing images and grayscale conversion can help the input, and convolutional neural networks can extract relevant features. Additionally, training stability was improved by adding techniques like experience replay and target networks. These approaches enable AI to learn and adapt its capabilities to effectively master Flappy Bird.

The Flappy Bird project provided several key learnings and insights emerged. We gained a deeper understanding of reinforcement learning principles, particularly how an agent can learn optimal strategies through trial and error by balancing exploration and exploitation. Implementing deep Q-learning demonstrated the importance of feature extraction and the role of conventional neural networks in processing complex inputs. The use of gymnasium in OpenAI’s Gym gave a wide range of environments for testing and developing reinforcement learning algorithms. This allowed users to easily reset, act, and observe outcomes, this facilitates robust development and evaluation of AI models. The project provided the significance of training stability techniques like target networks in achieving consistent performance improvements. Overall, the experience with real-time feedback and model tuning gave us valuable insights of both theoretical and practical aspects of machine learning and AI advancements.

s