Bradley Johnson

ITAI 1378

Professor McManus

11 December 2024

Capstone Project

1. Environment Setup

Flappy Bird Game Environment

Graphics: The Flappy Bird game is a simple 2D game where you control a bird. The background is a blue sky with a few clouds, and the ground has obstacles in the form of pipes. The bird needs to fly between the pipes without hitting them.

Physics: The bird falls due to gravity unless you make it flap. When you make the bird flap, it moves up. If it touches the pipes or the ground, the game ends.

Scoring System: You get one point every time the bird successfully flies between two pipes. The score goes up as long as you keep passing pipes without crashing.

Tools and Libraries

PyGame: PyGame is a tool that helps you make 2D games like Flappy Bird. It’s great for creating simple games because it can handle graphics and game actions easily. We use PyGame because it’s simple and has everything we need to make the game run.

OpenAI Gym: OpenAI Gym is a tool used to train computer programs to play games. It helps set up the game in a way that a computer can understand, making it easier to teach the computer to play Flappy Bird.

Game Setup for AI Interaction

State Representation: The computer needs to understand the game to make good choices. We can represent the game by looking at the bird’s position, how fast it’s falling, and where the pipes are.

Action Space: The actions the computer can take are simple: it can either flap the bird to move up or do nothing and let the bird fall.

Reward System: The computer gets a reward when it successfully passes a pipe (like a +1 point). If the bird hits a pipe or the ground, it gets a penalty (like -1).

Preprocessing Game Frames

Before giving the game image to the computer, we need to simplify it:

* Resizing: We make the game image smaller so the computer doesn’t have to deal with too much information.
* Grayscale Conversion: We turn the image into black and white. This helps the computer focus on the important parts of the game, like the pipes and the bird.

2. Pre-trained Model Usage

Transfer Learning Concept and Benefits

Transfer learning is when a computer program uses knowledge from one task and applies it to a new one. It helps save time because the computer doesn’t have to learn everything from scratch. In this case, we can use a pre-trained model that has already learned about shapes and objects to help with the Flappy Bird game.

Pre-trained Model: MobileNetV2

I would use MobileNetV2, a pre-trained model, because it’s fast and good at recognizing objects. Since it has already been trained on thousands of images, it can easily spot the bird and pipes in the Flappy Bird game.

Modifying MobileNetV2 for Feature Extraction

We can modify MobileNetV2 by removing the parts of the model that are used for classifying objects and instead use it to find important features in the game, like where the pipes and the bird are. We only need the parts of the model that help it understand the image, not to classify it.

Challenges in Adapting MobileNetV2

One challenge is that MobileNetV2 was trained on more complex images, so it might not work perfectly with the simple graphics of Flappy Bird. To solve this, we could resize the game’s images and tweak the model so it works well with the game.

3. Reinforcement Learning Implementation

Basics of Reinforcement Learning

In reinforcement learning, a computer learns by playing a game and getting rewards or penalties. The main parts are:

* States: The computer’s understanding of what’s happening in the game (like the bird’s position).
* Actions: What the computer can do, like flapping or doing nothing.
* Rewards: If the computer does something good, it gets a reward; if it does something bad, it gets a penalty.

Algorithm: Deep Q-Learning (DQN)

The computer will use Deep Q-Learning to learn how to play. This is a method where the computer guesses what will happen if it makes a certain choice, and it learns from those guesses over time.

Components of DQN

* Q-Network Architecture: This is like a brain for the computer. It looks at the current state (the bird and pipes) and decides which action (flap or stay still) will get the best outcome.
* Replay Memory: This stores the computer’s past actions so it can learn from them.
* Target Network: This helps make sure the computer’s guesses are stable and don’t change too quickly.

Exploration vs. Exploitation

At first, the computer will explore the game a lot, trying different actions. As it gets better, it will exploit its knowledge and choose the actions that give the best results.

Experience Replay

Experience replay helps the computer remember and learn from past experiences. This makes training faster and better because the computer can learn from a variety of different situations.

4. Model Training

Training Process

The training process involves letting the computer play the game many times. After each game, the computer looks at its choices and learns from them to make better decisions next time.

Training Loop Setup

The computer will go through many games, updating its brain (Q-network) after each action. This process will continue until the computer gets better at the game.

Hyperparameter Tuning

Some settings, called hyperparameters, affect how the computer learns. These include the learning rate (how quickly it learns), the discount factor (how much it cares about future rewards), and how fast it shifts from exploring to exploiting.

Handling Training Issues

Some problems that may come up are:

* Catastrophic Forgetting: The computer might forget what it learned earlier. Using experience replay can help solve this.
* Reward Sparsity: The computer might not get enough rewards, so we can give smaller rewards for things like surviving longer to help it learn faster.

Evaluating Performance

We can measure the computer’s performance by looking at how long it survives in the game and how many points it scores on average.

5. Testing and Evaluation

Testing Strategy

We will test the computer by letting it play many games and see how well it performs. We want to see if it can consistently pass pipes and avoid crashes.

Metrics for Performance

We will use average score (how many pipes the computer passes) and survival time (how long the computer survives) to evaluate how well it plays.

Interpreting Results

If the computer scores high points and survives for a long time, that means it’s learned to play well. We can compare it to how well a human would do in the game to see how good the AI is.

Visualizing Performance

We can show the game being played with markers that highlight the actions the computer takes, like when it decides to flap.

Potential Improvements and Future Work

In the future, we could make the game harder by adding more obstacles or changing the pipes’ speed. We could also improve the computer’s brain to make it smarter and faster.

Bibliography

* Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). *Human-level control through deep reinforcement learning*. Nature, 518(7540), 529-533.
* Chollet, F. (2017). *Deep Learning with Python*. Manning Publications.
* <https://medium.com/analytics-vidhya/how-i-built-an-ai-to-play-flappy-bird-81b672b66521>
* <https://flappybird-ai.netlify.app/>