ShaRise Griggs Carlos Granillo Jonathan Smith
ITAI 1378 Computer Vision
FInal Exam
Professor McManus

Flappy Bird AI Agent: Conceptual Report

Overview

In this project, the goal was to create an AI model that can play Flappy Bird by using reinforcement learning (RL) and transfer learning techniques. The main idea was to set up the game environment, have the AI learn how to play it through trial and error, and solve common challenges like sparse rewards and optimizing for efficiency. The project focused on using pre-trained models to help the AI recognize important features in the game, and then applying deep Q-learning (DQN) to help it make smart decisions.

How the Flappy Bird Game Works

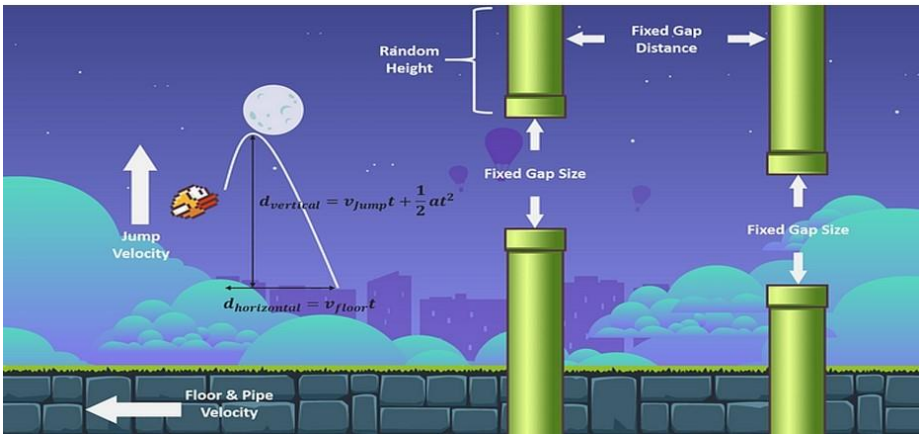
Flappy Bird is a simple 2D game where you control a bird that has to fly through a series of pipes. The bird is pulled down by gravity, and you can make it fly upward by tapping. You get points by passing through the gaps in the pipes, and the game ends if the bird hits a pipe or falls to the ground.

What Makes the Game Tick:

- **Graphics**: The bird is controlled in a 2D environment with a scrolling background and randomly placed pipes.
- **Physics**: The bird falls because of gravity, but tapping makes it rise. It's a simple, fun mechanic.
- **Scoring**: The more pipes the bird passes without hitting anything, the more points it gets. The game ends when the bird crashes.

Players earn points by guiding the bird through the openings between pipes. The current score is typically displayed at the top of the screen. The game ends when the bird collides with a

pipe, touches the ground, or hits the top of the screen ("Can Someone Explain Flappy Bird's Physics to Me?").



The Tools Used

To set up the game and train the AI, I used the following tools:

- **PyGame**: This Python library was used to create the game. It makes things like handling graphics, sound, and game logic easy to do.
- **OpenAI Gym**: This is a toolkit for creating and training reinforcement learning agents. It standardizes the way the AI interacts with the environment, making it easier to focus on the learning part.

What the AI Needs to Know

The AI needs to understand the current state of the game, so it can make decisions. The state includes:

- The bird's position and how fast it's moving up or down.
- How far the bird is from the next pipe and the size of the gap between pipes.

What the AI Can Do

The AI only has two options:

1. **Flap**: This makes the bird go up.
2. **Do nothing**: Let the bird fall.

How the AI Gets Feedback

- **Good Feedback**: The AI gets +1 for successfully passing a pipe.
- **Bad Feedback**: The AI gets -10 if it hits a pipe or falls to the ground.

Making the Game Frames AI-Friendly

The raw game frames need to be simplified so the AI can learn from them. Here's what I did:

1. **Resize**: The frames are resized to 84x84 pixels to make them smaller and easier to process.
2. **Convert to Grayscale**: The frames are changed to black and white to simplify them by reducing the number of colors.
3. **Normalize**: The pixel values are scaled between 0 and 1, which helps the AI handle the input better.
4. **Stack Frames**: I stack a few frames together so the AI can understand the bird's movement over time instead of just looking at one frame.

Using a Pre-Trained Model to Help the AI Learn

We used **transfer learning**, which is a technique where the AI uses a pre-trained model to learn faster ("What Is Transfer Learning?"). Instead of starting from scratch, the AI can reuse knowledge from a model that's already been trained on similar tasks. We chose **VGG16**, a model that's been trained on ImageNet. It's good at recognizing features in images, and by using this model, the AI can better understand the game frames without having to train from scratch.

ShaRise Griggs Carlos Granillo Jonathan Smith
ITAI 1378 Computer Vision
FInal Exam
Professor McManus

How We Adjusted the Model

1. **Freeze Layers**: The first layers of the pre-trained model were kept as they were so that the AI wouldn't forget what it had learned.
2. **Add Custom Layers**: I added layers at the end of the model to predict whether the bird should flap or do nothing.
3. **Final Output**: I used a softmax function to make sure the predictions add up to 100%, representing the probabilities for each action.

Challenges and How We Fixed Them

- **Different Game Frames**: The model was trained on images of everyday objects, not game frames, so it needed some adjustments. I solved this by adding data augmentation, like rotating and flipping the images to make the model more adaptable.

Reinforcement Learning: Teaching the AI to Learn from Its Actions

Reinforcement learning is like teaching the AI through trial and error. The AI learns by interacting with the game and receiving rewards or punishments based on its actions ("StyleSwin: Transformer-Based GAN for High-Resolution Image Generation.").

Deep Q-Learning (DQN)

DQN was used to make decisions based on what the AI sees. It uses a neural network to predict the expected rewards for each action the AI can take.

- **Q-Network**: The AI uses this network to figure out which action will give it the highest reward.
- **Replay Memory**: The AI stores its past experiences and learns from them later, which helps it avoid repeating the same mistakes.

- **Target Network**: This helps stabilize the learning process by updating the model's knowledge in intervals.

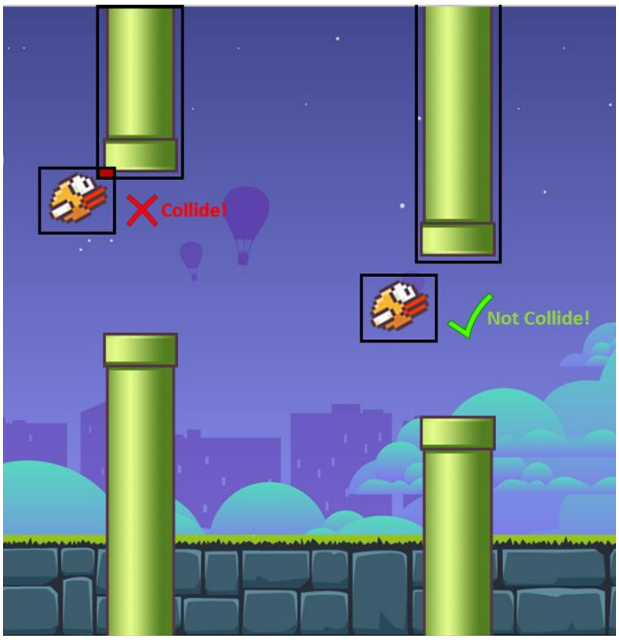
Exploration vs. Exploitation

At first, the AI tries random actions (exploration), but over time, it starts using the actions it knows will give the best results (exploitation). I used an **epsilon-greedy strategy** to balance this.

Experience Replay

Experience replay is used so the AI can learn from a variety of past experiences, rather than just the most recent ones. This helps it become more flexible in different situations.

Reinforcement Learning Implementation



Reinforcement learning is an effective method for training an AI agent to tackle complex games like Flappy Bird. The Deep Q-Network (DQN) algorithm is an advanced technique that

allows the agent to learn by interacting with the game, turning the task of maneuvering through obstacles into a well-defined learning process.

Testing the AI and Evaluating Its Performance

How We Tested the AI

We tested the AI by running several episodes and tracking how well it did. The key metrics we looked at were:

- **Average Score**: The number of pipes the AI passed on average.
- **Survival Time**: How long the AI survived before hitting a pipe or falling.

Results

We created graphs to show how the AI's performance improved over time. I also used heatmaps to visualize when the AI chose to flap or stay still.

What We Learned and Challenges I Faced

Key Insights

- **Exploration vs. Exploitation**: Balancing between trying new things and sticking to what works is crucial for the AI to improve. Too much exploration wastes time, but too much exploitation can make the AI stuck in a bad pattern.
- **Importance of Preprocessing**: Reducing the complexity of the game frames helps the AI learn faster and more effectively.

Challenges

- **Sparse Rewards**: It was hard for the AI to learn at first because it didn't get feedback often. To fix this, we gave intermediate rewards for staying alive longer.

- **Overfitting**: The model was at risk of learning too much from specific frames.

We solved this by using techniques like experience replay and target networks.

Next Steps and Improvements

1. **Advanced Learning Methods**: we could try using advanced RL methods like Proximal Policy Optimization (PPO) to make the learning process smoother.
2. **Dynamic Difficulty**: Adding dynamic difficulty could help make the game harder as the AI improves, providing a more challenging learning experience.
3. **Optimizing the Model**: we could use techniques like pruning or quantization to make the model faster and more efficient.

What We Learned Throughout This Project

Challenges Faced

- Making sure the AI could understand the game mechanics and respond appropriately took a lot of trial and error. It was tricky to get the AI to learn effectively because of the sparse feedback.
- Picking the right pre-trained model was important, and we had to adjust it to work well with Flappy Bird's simple but unique environment.
- Using pre-trained models saved a lot of time and helped the AI learn much faster.
- The reinforcement learning approach worked well once we addressed the exploration-exploitation issue and started tuning the AI's training.

ShaRise Griggs Carlos Granillo Jonathan Smith
ITAI 1378 Computer Vision
FInal Exam
Professor McManus

Works Cited

Sweeney, Forest. “Can Someone Explain Flappy Bird's Physics to Me?” Game Development Stack Exchange, 12 Jan. 2014, gamedev.stackexchange.com/questions/70268/can-someone-explain-flappy-birds-physics-to-me. Accessed 10 Dec. 2024.

Amazon Web Services. “What Is Transfer Learning?” Amazon Web Services, aws.amazon.com/what-is/transfer-learning/. Accessed 10 Dec. 2024.

Zhang, Han, et al. “StyleSwin: Transformer-Based GAN for High-Resolution Image Generation.” OpenReview, 25 May 2023, openreview.net/forum?id=T6RygOFZ6B. Accessed 10 Dec. 2024.