Capstone Project: Train an AI Agent to Play Flappy Bird

Martin Demel

Department of Science, Technology, Engineering & Math, Houston Community College

ITAI-1378: Computer Vision Artificial Intelligence

Patricia McManus

December 11th, 2024

Introduction

This assignment demonstrates the ability to train an AI agent to play Flappy Bird using computer vision techniques and reinforcement learning. To teach the agent to read from raw pixels, my initial focus was on setting up the environment correctly and integrating a pretrained model for feature extraction.

During this journey, I have encountered several challenges that I will cover in my report. Furthermore, I have experimented with tweaks to hyperparameters and techniques and even attempted to implement advanced integration with external tools, including W&B and Optuna, to demonstrate my research initiative and personal curiosity about improving the model.

Even though the final results were not satisfactory, I learned how to modify the environment as the project progressed, including changes to model architecture and tuning hyperparameters.

Project Structure and Definition

To ensure that the project is defined with a focus on clarity and visibility, I have created five modules to define the project's structure clearly:

1. Environment Setup
2. Pre-trained Model Integration
3. Reinforcement Learning Implementation (DQN)
4. Model Training
5. Testing and evaluation

This structure will ensure that each module will be clearly defined and follow the systematic approach I have taken to conduct this project.

1. Environment Setup

I installed the necessary libraries to initiate the project, including NumPy, TensorFlow, Keras, gym, ple, and OpenCV-python. I also initialized the PyGame Learning Environment (PLE) to run the Flappy Bird Game, which required changes to my virtual Python environment's kernel.

I have considered running the project in a cloud environment, but ultimately decided to perform the tasks on my local MacBook Pro M3 Max, leveraging Apple's tensorflow-metal integration to enable GPU acceleration. I encountered initial version mishmash, and I had to ensure I updated my kernel to a supported version by installing the correct Python version in Terminal.

After importing the necessary libraries and ensuring my environment is ready to run the project, I implemented the preprocessing function to resize the frames to 224 by 224 and normalize pixel values. This ensured compatibility with MobileNet V2, the pre-trained model I chose because of its efficiency and decent performance. I used the Flappy Bird environment without any visual display to speed up the training process by defining Display_Screen=False.

2. Pre-trained Model Integration

Once the environment was set, I integrated the actual MobileNet V2 by removing the top layers and freezing its parameters to retain its learned visual features. Furthermore, I placed a dense layer to ensure we could correctly predict the Q-values for the Flappy Bird game agents' actions. I have started with the simple linear output layer; however, to address the poor performance I was experiencing, I experimented with adding an intermediate Dense Layer, 128 Units\ ReLU, to increase the representational capacity of the network. This adjustment was just one of the experiments to improve policy learning, which ultimately did not lead to a performance boost. However, I have learned new

techniques that helped increase my knowledge and explore new ways of handling the challenges of training the agent.

3. Reinforcement Learning Implementation (DQN)

Due to the complexity of the problem and the advice in the instructions we received as part of this project, I have decided to utilize the Deep Q-Network. I have also decided to include Experience replay to break the correlation between the consecutive states, an epsilon-greedy policy with an epsilon decay schedule, and a target network with periodic updates to stabilize the Q-Learning.

During the testing and fine-tuning, I have tried different epsilon decay rates, including 0.995 & 0.999, memory sizes varying from 50,000 to 100,000, as well as different batch sizes. I was hoping to see significant improvements in strategy by the agent. I have also tested various reward methods, starting with a small survival reward to a drastically increased reward to + 0.1 without success.

4. Model Training

Initially, the agent would always score -5.0 in the test episodes, which meant that the bird crashed immediately after the start. When I changed the reward and slowed the epsilon decay, I occasionally saw episodes improve with slightly better average scores of -4.0, which gave me an average of -4.8.

Furthermore, to improve overall performance, I have tried to adjust the frame skipping to speed up the training. I have also increased or decreased the number of epochs, ranging from 50 to 200. Overall, more episodes allowed the agent more opportunities to learn, but even with 200 epochs, no dramatic improvement was observed, as seen in Figure 1—Model Training performance.

Research initiative—W&B and Optuna

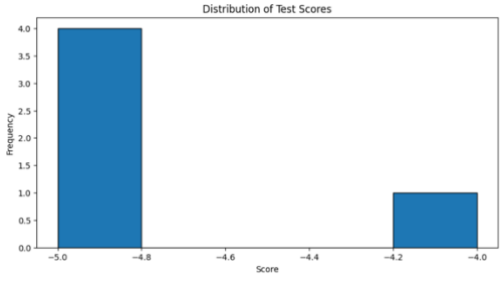
Since the results were not satisfactory, I have decided to perform a dramatic change to the architecture.

I have attempted to use Weights and Biases for advanced login and visualization and to implement Optuna for systematic hyperparameter optimization. Unfortunately, while these techniques and tools improved my knowledge and helped me explore deeper, I was not successful in significantly improving the results. The results remained close to -5.0 on average. One test epoch reached -4.0, which was still a minor improvement given the advanced methodology.

Nevertheless, this experiment taught me how the weights, biases, and Optuna can be implemented for agent graining.

5. Testing and Evaluation

I have summarized the testing by setting epsilon=0.0 to determine the testing over several episodes. To further evaluate the results, I have printed out each test episode's score and calculated the average as the final score. As the data was present, I plotted a line graph of the raw training scores to observe how performance evolved over episodes. Furthermore, to show the trend clearly, I included the rolling average to smoothen the graph. Finally, a histogram with the test scores is used to visualize the distribution at the end of training. This can be seen in Figures 2, 3, and 4—Training scores over episodes and distribution.



Personal Reflection

By performing this project, I have learned new ways to reach the agent. It reminded me of our previous assignment on what Alphastar cannot do. This hands-on experience showed me the importance of selecting the right tools and techniques and fine-tuning the parameters. I have also learned that training Agents directly from pixels can be very challenging, especially in environments like Flappy Bird, especially when I am not able to adjust the game difficulty and parameters.

This brings me to the point that during my project, I conducted online research and found an article by Zhu (2021) describing the process of building an intelligent agent to play Flappy Bird. After reading the article, I tried to apply some of the techniques to my project, including directly using visual inputs, reward shaping to improve survival, longer training time, and parameter tuning.

This article was eye-opening, and I really enjoyed reading it. It reminded me of the hidden layers in neural networks and taught me a little more about NEAT—NeuroEvolution of Augmenting Topologies.

Conclusion

Overall, I have very much enjoyed the project, including the research part, which taught me some advanced techniques that I have used in the project. Even though the final results and performance were only slightly improved, this project deepened my experience and understanding of the environment, setting up the parameters, integrating pre-trained models, and performing advanced logging and optimizations.

It has also deepened my knowledge of my work experience. When I work on digital transformation solutions, I often interact with chatbots and AI agents. Although this project focuses on Vision of Pixels, the reinforcement learning principles are similar.

References:

Zhu, D. (2021, December 14). How I built an intelligent agent to play Flappy Bird. Medium. https://medium.com/analytics-vidhya/how-i-built-an-ai-to-play-flappy-bird-81b672b66521