Jeffery Dirden

Bradley Johnson

Professor McManus

December 5, 2024

Capstone Project: Train an AI Agent to Play Flappy Bird

For this project we decided to use the conceptual to explain and cover how we would approach this challenge. Flappy Bird is a 2D side-scrolling game with simple but challenging mechanics. The player takes control of a bird that must navigate a series of vertical pipes with varying gap sizes while avoiding collisions. When no action is taken, gravity pulls the bird downward and it moves upward with a flap. This results in a dynamic and unforgiving gameplay experience where timing and precision are essential. The game's scoring system is simple: the player receives one point for each pipe successfully passed, and the game ends when they collide with a pipe or the ground. Flappy Bird is an ideal candidate for AI experimentation because of its simplicity and high skill requirement.

To simulate the Flappy Bird environment for AI interaction, I would use PyGame and OpenAI Gym. PyGame is a powerful library for creating and customizing 2D games, with simple tools for rendering graphics and controlling physics. OpenAI Gym enhances PyGame by offering a standardized interface for integrating reinforcement learning algorithms. This combination

creates a cohesive and flexible environment for developing, testing, and optimizing the AI agent. The setup for AI interaction necessitates precise definitions of state representation, action space, and reward system. The state representation should include important information about the environment, such as the bird's vertical position, velocity, distance to the next pipe, and gap location. These variables are essential for the AI agent to make sound decisions. The action space in Flappy Bird is binary: the bird can flap upward or do nothing, allowing gravity to pull it downward. The reward system allows for a positive reward (+1) for successfully passing a pipe, while collisions result in a penalty (-1). Furthermore, a small survival reward (+0.1) for each frame can encourage longer gameplay by balancing short-term rewards and long-term survival. Preprocessing techniques are required to prepare game frames for artificial intelligence input. Frames are resized to a lower resolution, such as 84x84 pixels, to reduce computational complexity while preserving critical spatial information. Grayscale conversion is used to simplify the input by removing unnecessary color information, and pixel values are normalized to fall between 0 and 1. Frame stacking combines consecutive frames to improve the agent's ability to perceive movement and predict future states.

Transfer learning is a machine learning technique that adapts a previously trained model to solve a new, related problem. Because the model draws on knowledge from previous tasks, this method significantly reduces training time and computational resources. Transfer learning can be used to process visual input in Flappy Bird, specifically to recognize patterns in the game environment such as pipes, gaps, and the bird's position. For this project MobileNetV2, with its lightweight architecture and efficiency, is an appropriate pre-trained model for this project. MobileNetV2, pre-trained on the ImageNet dataset, excels at extracting visual features like edges

and shapes. These characteristics are directly applicable to Flappy Bird, whose environment consists of distinct geometric structures. MobileNetV2 is modified for this task by replacing its final classification layers with custom layers designed to predict Q-values for actions in the Flappy Bird environment. To preserve learned features, the model's initial layers are frozen, while the later layers are fine-tuned to adapt to the game dynamics. Despite its benefits, adopting a pre-trained model poses several issues. Most pre-trained models are meant to classify static images, whereas Flappy Bird uses dynamic, sequential frames. This constraint can be overcome by combining transfer learning and frame stacking, which provide the model with the temporal context required for decision-making. These issues can be mitigated by successfully preprocessing the game frames and fine-tuning the model, allowing the AI agent to process visual information more efficiently.

Reinforcement learning (RL) is a machine learning paradigm in which an agent learns to make decisions by interacting with its surroundings and optimizing cumulative rewards. The RL framework includes states, actions, rewards, and policies. In the Flappy Bird game, the status indicates the bird's vertical position, velocity, distance to the next pipe, and gap location. The action space is binary, with the options being "flap" or "do nothing." Rewards are given based on performance, such as surviving longer or going through pipes. Deep Q-Learning (DQN) is a widely used RL method for applications with discrete action spaces. It estimates Q-values for state-action pairs using a neural network known as the Q-network. The Q-value shows the expected cumulative benefit for carrying out a specified action in a given state. DQN has two major innovations: experience replay and a target network. Experience replay entails storing previous interactions in a replay buffer and sampling random batches while training. This

disrupts temporal connections and increases learning stability. The periodically updated target network provides steady Q-value targets during training, decreasing oscillations and divergence. Balance between exploration and exploitation is crucial in RL. An ε-greedy strategy achieves this balance by picking random actions ("exploration") with a probability ε and the highest Q-value ("exploitation") otherwise. The value of ε lowers over time, ensuring that the agent investigates enough during early training and applies its learnt policy in later phases. Experience replay is critical for effective learning. By sampling varied experiences, the agent avoids overfitting to recent interactions and generalizes more effectively across different gameplay contexts. This technique, when combined with DQN's architecture, ensures robust and quick learning, allowing the AI agent to navigate the Flappy Bird environment well.

The model training procedure starts with initializing the environment and the Q-network. During each episode, the agent interacts with the environment, observes the current state, chooses an action based on the ε-greedy policy, and executes it. The environment responds by transitioning to a new state and giving the agent a reward, which it uses to update the Q-network. This encounter is saved in the replay buffer, allowing the agent to sample different sets of experiences during training. The training loop has numerous steps. First, a sample of experiences is taken from the replay buffer. The Q-network predicts Q-values for current states and calculates goal Q-values based on rewards and future states. The network's parameters are adjusted using gradient descent to reduce the difference between predicted and target Q-values. The target network is periodically modified to align with the Q-network, ensuring that learning remains stable. Hyperparameters like learning rate, discount factor (γ), replay buffer size, batch size, and ε-decay rate all have a substantial impact on training. A high learning rate, for example, might

accelerate training but cause instability, whereas a low rate ensures consistent improvement. The discount factor assesses the significance of future benefits by weighing short-term and long-term profits. These hyperparameters can be optimized via grid search or iterative testing. Common training obstacles include catastrophic forgetting, in which the agent forgets previously learned information, and reward sparsity, in which rewards are few. Catastrophic forgetting can be addressed with a target network, and reward sparsity can be reduced by incorporating intermediate rewards, such as survival rewards, to direct the agent's learning. Evaluating the model's performance during training entails tracking parameters such as average score and survival time across episodes, which provide insights into the agent's progression and opportunities for development.

The trained agent is tested by running it through multiple episodes to evaluate its performance in a variety of settings. A comprehensive testing technique includes evaluating the agent's ability to negotiate difficult areas and adjust to changes in the game's dynamics. Metrics such as average score, maximum score achieved, and average survival time are used to evaluate the agent's performance. against interpret the results, compare the agent's performance against benchmarks like a random agent or a human player. This comparison focuses on the AI agent's strengths and flaws, providing useful insights into its decision-making abilities. Visualization tools, such as plots of average scores over time and gaming recordings, can help to illustrate the agent's behavior and progress throughout training. Based on the evaluation results, prospective enhancements include experimenting with sophisticated RL algorithms such as Proximal Policy Optimization (PPO) or introducing new characteristics into the state representation, such as pipe velocities. Future research could potentially look into curriculum learning, in which the agent

begins with basic versions of the game before progressing to more complex scenarios. These enhancements would allow the AI agent to perform and adapt more effectively, showcasing the strength and flexibility of reinforcement learning.

In conclusion training an AI agent to play Flappy Bird using reinforcement learning and computer vision is a difficult but gratifying task. The AI agent can attain exceptional performance by methodically preparing the environment, utilizing pre-trained models, implementing powerful RL algorithms, and fine-tuning the training process. This study illustrates AI's potential for handling dynamic and complicated challenges, setting the path for future research and innovation in the field of artificial intelligence.

Bibliography

1. Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd Edition). MIT Press.
    - This textbook provides foundational knowledge on reinforcement learning concepts, including algorithms like Q-learning and deep Q-networks.
2. Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). "Human-level control through deep reinforcement learning." Nature, 518(7540), 529-533.
    - This paper introduced Deep Q-Networks (DQN), a critical algorithm used in training agents to play games like Flappy Bird.
3. OpenAI Gym Documentation
    - Available at: https://www.gymlibrary.dev
    - Provides an overview of tools and environments for developing and comparing reinforcement learning algorithms.
4. PyGame Documentation
    - Available at: https://www.pygame.org/docs

- Details about the PyGame library, which can be used to create and simulate the Flappy Bird environment.

5. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
   - Covers deep learning models and their applications, including convolutional neural networks (CNNs) relevant to preprocessing game frames.

6. He, K., Zhang, X., Ren, S., & Sun, J. (2016). "Deep Residual Learning for Image Recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
   - Explains the architecture of ResNet, a pre-trained model that can be adapted for feature extraction in Flappy Bird.

7. Hasselt, H. V., Guez, A., & Silver, D. (2016). "Deep reinforcement learning with double Q-learning." Proceedings of the AAAI Conference on Artificial Intelligence.
   - Provides insights into Double Q-learning, an improvement over traditional Q-learning to reduce overestimation biases.

8. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). "ImageNet Classification with Deep Convolutional Neural Networks." Advances in Neural Information Processing Systems (NeurIPS).
   - Discusses the development and training of deep convolutional neural networks like AlexNet, useful for understanding preprocessing and feature extraction.

9. Andrychowicz, M., Denil, M., Gomez, S., et al. (2016). "Learning to learn by gradient descent by gradient descent." Advances in Neural Information Processing Systems (NeurIPS).
   - Explores optimization methods that are foundational for fine-tuning neural networks in reinforcement learning.

10. YouTube Tutorials on Flappy Bird AI
    - Various tutorials demonstrating the implementation of reinforcement learning for Flappy Bird, including channels like "Sentdex" and "Code Bullet."

11. Medium Articles on Deep Q-Learning and Game AI
    - Examples: "Reinforcement Learning for Flappy Bird" by Daniel Slater and "A Beginner's Guide to Deep Q-Learning" by Towards Data Science.

12. GitHub Repositories on Flappy Bird RL
    - Examples: https://github.com/yenchenlin/DeepLearningFlappyBird
    - Open-source projects that demonstrate how to implement RL for Flappy Bird.

13. Microsoft Azure Machine Learning Documentation
    - Available at: https://learn.microsoft.com/en-us/azure/machine-learning

- Resources on leveraging cloud-based solutions for training and deploying reinforcement learning models.

14. NVIDIA Deep Learning Documentation

    - Available at: https://developer.nvidia.com/deep-learning