Cassy Cormier

Learning Log: Train an AI Agent to Play Flappy Bird

15698 Computer Vision 1378

Professor Anna Devarakonda

April 20th, 2025

Learning Log: Flappy Bird AI Agent Project

- **Challenges Faced & Solutions Implemented**

1. Environment Setup Failures
    a. Early attempts to run the Flappy Bird environment in Google Colab often resulted in broken runtimes and execution errors.
    b. Solution: Restarting the Colab runtime session consistently helped resolve these errors. Eventually, setting up the environment locally with editable package installation improved reliability.
2. 2. Epsilon Decay & Exploration Strategy Flaws
    a. Challenge: In Test 1, the agent's epsilon value was stuck at 1.0, meaning it kept making random moves even after 500 episodes. No learning or rewards were recorded.
    b. Solution: The epsilon decay schedule was rechecked and implemented correctly, leading to a gradual shift from exploration to exploitation. By Test 4, epsilon had decayed to 0.1 and the agent showed improved behavior.
3. Sparse Reward Signal
    a. Challenge: With the default reward system, the agent mostly received -5.0 for dying and almost never survived long enough to get positive feedback.
    b. Solution: Introduced survival rewards (+0.1 per frame) to provide denser feedback and allow the agent to learn incrementally. This drastically improved learning stability.
4. Coding Bugs & Debugging Under Pressure
    a. Challenge: Frequent bugs in the step function and inconsistent environment behavior made debugging difficult, especially with time constraints.
    b. Solution: Leveraged ChatGPT and ClaudeAI for live debugging help, which saved time and supported deeper understanding of Python and ML code structures.
5. No Evidence of Learning
    a. Challenge: In early tests, the agent showed no meaningful improvement in performance.
    b. Solution: Introduced logging, performance tracking, and a reward plot function. Saved best-performing models automatically to monitor learning progress.

- **Key Learnings & Insights**

1. On Reinforcement Learning
    a. Gained firsthand experience with Deep Q-Learning (DQN), epsilon-greedy strategies, and how crucial reward design is for agent performance.
    b. Learned the impact of exploration-exploitation balance, reward sparsity, and target network updates in stabilizing training.
2. On Computer Vision

   a. Used image preprocessing, normalization, and grayscale conversion to reduce complexity and focus learning on relevant features like pipe edges and gaps.
   b. Applied feature extraction and object detection techniques to enhance the agent's decision-making capability.
3. Personal Growth
   a. Discovered patience and persistence are key in machine learning experiments. Debugging, especially when rewards plateau, taught resilience.
   b. Found joy in seeing meaningful results after solving a challenging problem—like when the agent first survived multiple pipes in Test 3.
4. Improvements to Consider
   a. Optimization Ideas
      i. Incorporate hyperparameter tuning to optimize learning rate, discount factor, and batch size.
      ii. Try Prioritized Experience Replay to speed up learning from high-value experiences.
   b. Technical Enhancements
      i. Explore Transfer Learning by fine-tuning pre-trained networks like MobileNetV2 for different environments.
      ii. Add multi-agent training to simulate cooperation or competition.
5. Game Environment Enhancements
   a. Introduce dynamic difficulty adjustment, like increasing pipe speed or spacing over time. Add more feedback types—like small positive rewards for flapping without dying—to reduce sparse rewards early on.

- Resources Used

1. ChatGPT (OpenAI): Assisted with Python debugging and reinforcement learning concept clarification.
2. ClaudeAI (Anthropic): Provided additional help with environment setup and code optimization.
3. MobileNetV2 (TensorFlow/Keras): Used as a pre-trained model for feature extraction in the DQN agent.
4. PyGame: Game engine for simulating the Flappy Bird environment.
5. OpenCV: Used for image preprocessing (grayscale conversion, normalization, feature extraction).
6. TensorFlow/Keras: Built and trained the neural network models.
7. JupyterLab & Google Colab / Local Setup: Used for coding, debugging, and model training.