Reginald Jones
February 3, 2026
Artificial Intelligence Assignment 3: Computer Science 311

2. Name and Purpose of the Application

- **Name:** CP-DQN Balancer.
- **Purpose:** This agent is designed to solve the "CartPole-v1" simulation. The goal is to keep a pole balanced vertically on a moving cart for as long as possible. It serves as a practical application of how an agent can learn optimal control policies in a continuous state space through trial and error.

3. Algorithms Used

- **Deep Q-Network (DQN):** This was chosen because traditional Q-learning requires a discrete "table" of states. Since the cart's position and velocity are continuous numbers, a Neural Network (DQN) is required to approximate the best actions.
- **Key Concepts:** The agent uses **Experience Replay** (to learn from past mistakes) and **Target Networks** (to stabilize training).

4. Dataset Information

- **Source:** Gymnasium "Classic Control" suite.
- **Records:** Not applicable in the traditional sense, as data is generated "on-the-fly" during 10,000 training timesteps.
- **Features:** 4 continuous features:
    1. Cart Position (-4.8 to 4.8)
    2. Cart Velocity (-Inf to Inf)
    3. Pole Angle (~ -0.418 rad to 0.418 rad)
    4. Pole Angular Velocity (-Inf to Inf)
- **Preprocessing:** The Gymnasium environment standardizes these values automatically for the neural network.

5. Libraries, Toolkits, and Frameworks

- **Gymnasium:** For the simulation environment.
- **Stable Baselines3:** For the pre-implemented DQN algorithm.
- **PyTorch:** As the deep learning engine for the neural network.
- **NumPy:** For data array manipulation.

6. Application Design and Implementation

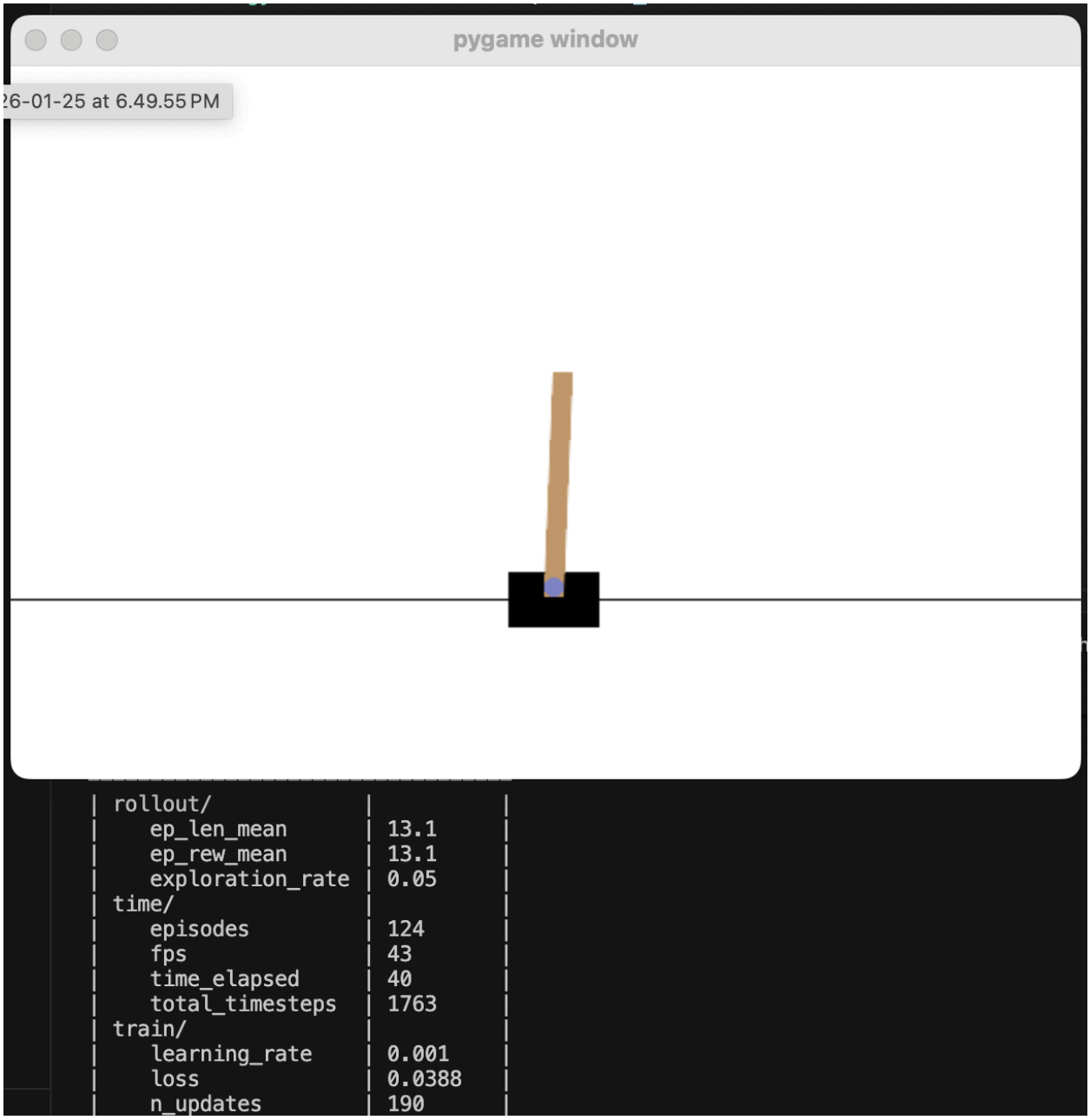
The **Agent** observes the **Environment** (State), chooses an **Action** (Left/Right) based on its **Policy**(DQN), receives a **Reward** (+1), and updates its internal model to improve future decisions.

7. Instructions for Running the Agent

1. Ensure Python 3.8+ is installed.
2. Place `app.py` and `requirements.txt` in a folder.
3. Run `pip install -r requirements.txt`.
4. Run `python agent.py`.
5. Observe the training logs in the console and the visual simulation window.

8. Results

- **Performance:** The agent achieved a "solved" status (usually a score of 500) within the 10,000 steps.

- 

9. Discussion and Insights

- **Performance:** The DQN is highly effective for this task due to its ability to generalize across continuous states.
- **Limitations:** The model can be sensitive to "hyperparameters" like the learning rate.
- **Improvements:** Future iterations could use **PPO (Proximal Policy Optimization)** for even faster training stability.

10. References (APA Style)

- Brockman, G., et al. (2016). *OpenAI Gym*. arXiv preprint arXiv:1606.01540.
- Raffin, A., et al. (2021). *Stable Baselines3: Reliable Reinforcement Learning Implementations*. Journal of Machine Learning Research.