Artificial Neural Networks - Fifth Assignment

Kiana Ghamsari 400222079

Report on Training a Reinforcement Learning Agent for BipedalWalker-v3

Introduction

The primary objective of this project was to train a reinforcement learning agent to master the BipedalWalker-v3 environment using the Proximal Policy Optimization (PPO) algorithm. The project involved several key steps: building and training the PPO agent, evaluating its performance, and saving and visualizing the best-performing episode.

Methodology

1. Environment Setup

The BipedalWalker-v3 environment was set up using the `gym` library. This environment is well-suited for continuous control tasks as it provides a continuous state and action space. The state space consists of various sensor readings from the bipedal walker, and the action space consists of the torques applied to the walker's joints.

- Action Space

  Actions are motor speed values in the [-1, 1] range for each of the 4 joints at both hips and knees.

- Observation Space

  State consists of hull angle speed, angular velocity, horizontal speed, vertical speed, position of joints and joints angular speed, legs contact with ground, and 10 lidar rangefinder measurements. There are no coordinates in the state vector.

- Rewards

  Reward is given for moving forward, totaling 300+ points up to the far end. If the robot falls, it gets -100. Applying motor torque costs a small amount of points. A more optimal agent will get a better score.

- Starting State

  The walker starts standing at the left end of the terrain with the hull horizontal, and both legs in the same position with a slight knee angle.

- Episode Termination

  The episode will terminate if the hull gets in contact with the ground or if the walker exceeds the right end of the terrain length.

Action Space	`Box(-1.0, 1.0, (4,), float32)`
Observation Space	`Box([-3.1415927 -5. -5. -5. -3.1415927 -5. -3.1415927 -5. -0. -3.1415927 -5. -3.1415927 -5. -0. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. ], [3.1415927 5. 5. 5. 3.1415927 5. 3.1415927 5. 5. 3.1415927 5. 3.1415927 5. 5. 1. 1. 1. 1. 1. 1. 1. 1. 1. ], (24,), float32)`
import	`gymnasium.make("BipedalWalker-v3")`

2. Neural Network Architecture

The PPO agent is comprised of two main components: the policy network and the value network. Both networks share a similar architecture with three fully connected layers.

Policy Network

- **Input Layer:** Takes the state representation from the environment.
- **Hidden Layers:** Two hidden layers with 256 neurons each, using ReLU activation functions.
- **Output Layer:** Outputs the mean of the action distribution and the log standard deviation of the action distribution.

Value Network

- **Input Layer:** Takes the state representation from the environment.
- **Hidden Layers:** Two hidden layers with 256 neurons each, using ReLU activation functions.
- **Output Layer:** Outputs a single value representing the estimated return (value) of the input state.

Training Setup

- **Optimization:** Both networks are optimized using Adam optimizer.
- **Discount Factor (γ):** 0.99, which balances immediate and future rewards.
- **PPO Clip Parameter (ε):** Limits the changes in the policy updates to improve training stability.
- **Training Epochs (K):** Number of epochs to update the networks per iteration.

Detailed Steps and Implementation

1. **Initialize Environment and Agent:**
    - The BipedalWalker-v3 environment is initialized.
    - PPO agent is created with state and action dimensions matching the environment.
2. **Training Loop:**
    - For each episode, the agent interacts with the environment to collect experiences (states, actions, rewards).
    - At the end of each episode, the collected experiences are used to update the policy and value networks.
    - The reward for each episode is recorded to track the agent's performance over time.

3. **Policy Update:**
    - At regular intervals, the agent's policy is updated using the collected experiences.
    - The policy update involves calculating the advantages and optimizing the surrogate objective function with clipping to ensure stable updates.
4. **Evaluation:**
    - The agent's performance is evaluated periodically by calculating the cumulative rewards over several episodes.
    - The best-performing episode is identified and saved for visualization.

3. PPO Agent

The PPO agent interacts with the environment and updates its policy based on the experiences collected. Key hyperparameters for the PPO algorithm included:

- Learning rate
- Discount factor (gamma)
- Clipping parameter (eps_clip)
- Number of epochs (K_epochs) for policy updates

The agent uses the collected experiences to optimize the policy and value networks. The PPO algorithm helps maintain a balance between exploration and exploitation by clipping the policy update to prevent drastic changes.

This chosen algorithm, Proximal Policy Optimization (PPO), which is a state-of-the-art reinforcement learning algorithm known for its stability and efficiency in training complex environments with continuous action spaces, such as the BipedalWalker-v3 environment.

PPO belongs to the family of policy gradient methods, where the agent learns a policy that directly maps states to actions. Unlike traditional methods that update the policy based on the entire trajectory, PPO updates the policy based on a clipped surrogate objective, which helps prevent large policy updates that could destabilize learning.

One key feature of PPO is its use of a clipped objective function, which limits the change in the policy during each update. This helps prevent the policy from deviating too far from its previous state, ensuring more stable learning. Additionally, PPO uses an adaptive clipping parameter that adjusts based on the current policy's performance, allowing for more effective policy updates.

PPO also incorporates an entropy term in its objective function, which encourages exploration by penalizing overly certain policies. This helps prevent the agent from getting stuck in local optima and encourages it to explore the environment more thoroughly.

- Ratio Function:

  In PPO, the ratio function calculates the probability of taking action a at state s in the current policy network divided by the previous old version of policy.

  In this function, rt(θ) denotes the probability ratio between the current and old policy:

If $r_t(\theta) > 1$, the action a at state s is more likely based on the current policy than the old policy.

If $r_t(\theta)$ is between 0 and 1, the action a at state s is less likely based on the current policy than the old policy.

This ratio function can easily estimate the divergence between old and current policies.

4. Training Procedure

The agent was trained over 3000 episodes. In each episode, the agent interacted with the environment, collected experiences, and updated its policy at regular intervals. The rewards for each episode were recorded to monitor the agent's performance over time.

5. Evaluation

After training, the agent's performance was evaluated over 10 episodes. The total reward for each episode was recorded, and the best-performing episode was identified and saved as a video for further analysis.

Results

Training Rewards

The rewards obtained during training demonstrated the agent's progress. Here are some sample rewards:

Episode	Reward
1	-101.27
2	-113.72
3	-125.93
...	...
100	-110.19
...	...
1000	134.47
...	...
2000	-104.07
...	...
3000	216.63

Best Episode:

- **Episode:** 2451
- **Reward:** 240.86

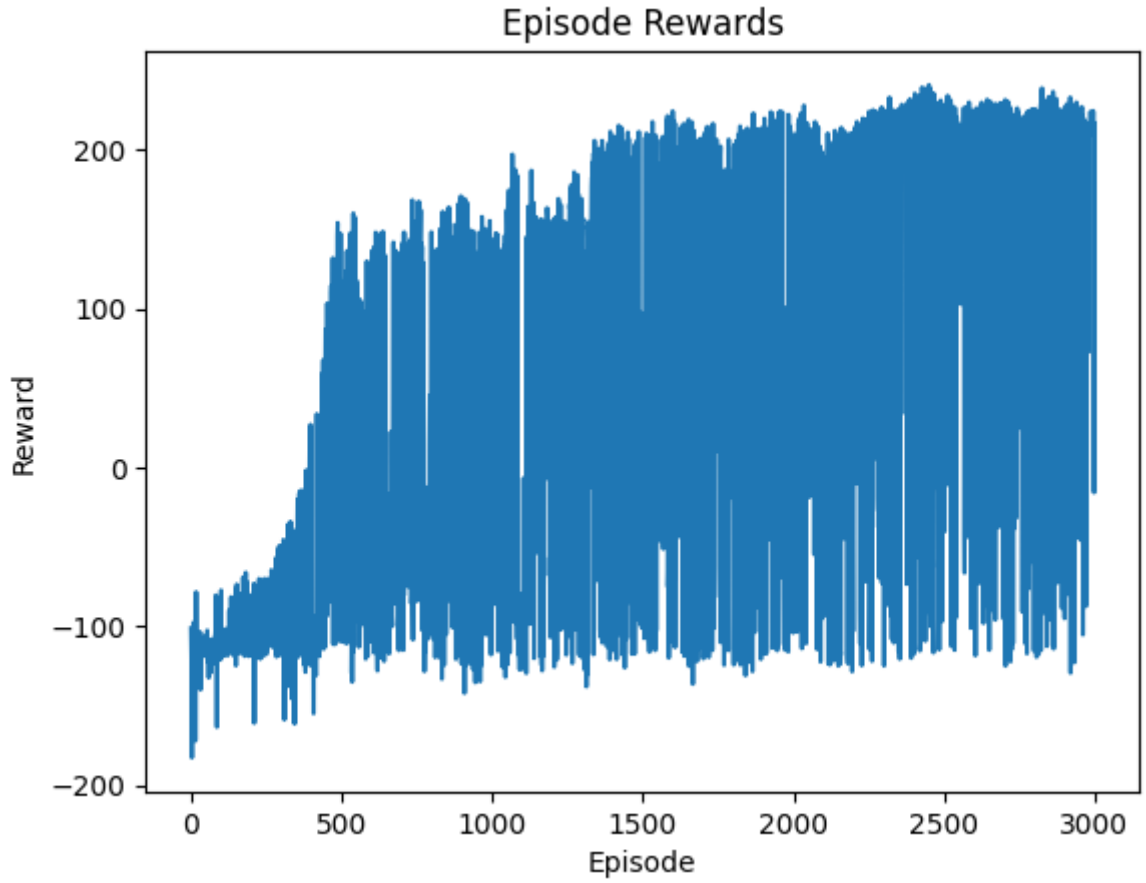
The rewards plot showed a significant improvement in the agent's performance over time, indicating successful learning and policy optimization.

Evaluation Rewards

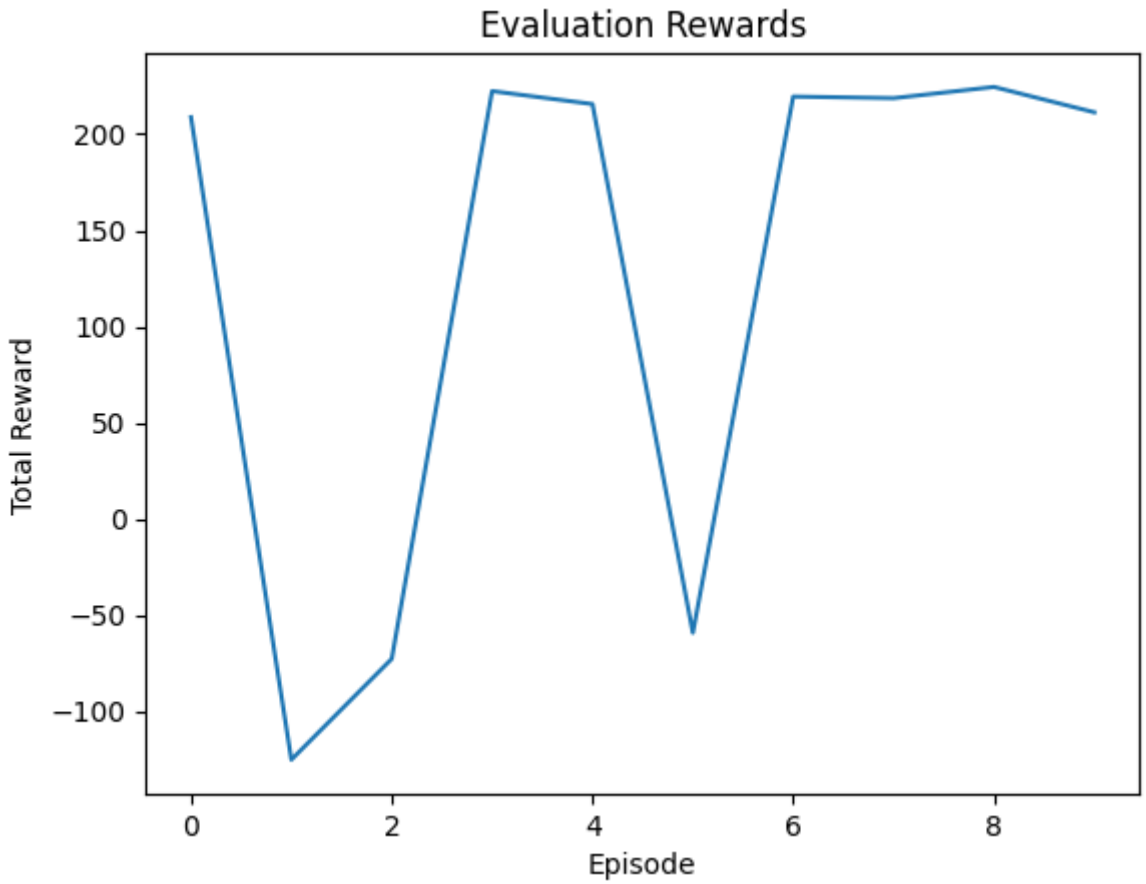
The agent's performance during evaluation was consistent with the training results.

Images and Visualizations

- **Episode Reward Plot:** Shows the cumulative reward per episode over the training period. This plot highlights the learning curve of the agent.



- **Evaluation Rewards Plot:** Illustrates the rewards obtained during the evaluation phase, demonstrating the agent's performance consistency.



- **Best Episode Video:** The video of the agent's best-performing episode is saved as `./video_eval/best_episode_episode_8.mp4`, showcasing the agent's ability to navigate the environment effectively.

Discussion

Performance Analysis

The agent's performance improved significantly over the training episodes, as evidenced by the increasing trend in the rewards. The best episode achieved a reward of 240.86, indicating that the agent learned effective policies to navigate the environment successfully.

Challenges and Solutions

1. **Exploration vs. Exploitation:** Balancing exploration of new actions with exploitation of known successful actions was challenging. This was mitigated by using a stochastic policy that encouraged exploration.
2. **Policy Stability:** Ensuring stable policy updates was crucial. The PPO algorithm's clipping mechanism helped maintain stability during training.

Potential Improvements

1. **Hyperparameter Tuning:** Further fine-tuning of the learning rate, discount factor, and PPO clip parameter could yield better performance.
2. **Network Architecture:** Experimenting with deeper or more complex network architectures might improve the agent's ability to learn more complex policies.

3. **Reward Shaping:** Modifying the reward structure to provide more informative feedback could accelerate learning.

Conclusion

The project successfully implemented a PPO agent for the BipedalWalker-v3 environment, achieving a significant improvement in performance over the training period. The use of PPO allowed for stable and efficient training, resulting in an agent capable of navigating the environment effectively. Future work could focus on further optimization and exploring advanced techniques to enhance the agent's capabilities.