Aryan Neizehbaz – 400222112
5nd Assignment (RL)

Environment

The BipedalWalker-v3 environment in OpenAI Gym simulates a bipedal robot that needs to learn to walk across a rugged landscape. This environment is particularly challenging due to the complexity of the robot's movements and the variability of the terrain.

- **Observations**: The state of the robot is represented by a 24-dimensional vector. This vector includes various physical properties such as the position, velocity, angle, and angular velocity of different parts of the robot.
- **Actions**: The robot's movements are controlled by a 4-dimensional continuous vector, which represents the torques applied to the robot's joints.

Algorithm Selection

For this task, an advanced RL algorithm suitable for continuous action spaces is required. Proximal Policy Optimization (PPO) is chosen due to its robustness and effectiveness in dealing with continuous control problems. Alternatives like Deep Deterministic Policy Gradient (DDPG) and Twin Delayed Deep Deterministic Policy Gradient (TD3) could also be considered, but PPO offers a good balance of simplicity and performance.

Implementation

The implementation involves using the Stable Baselines3 library, which provides pre-implemented versions of popular RL algorithms, including PPO. The key components of the implementation include:

- **Policy Networks**: Neural networks that decide the actions based on the current state.
- **Value Networks**: Neural networks that estimate the value of being in a given state.

- **Loss Functions**: Functions that help the agent learn by comparing predicted outcomes with actual outcomes.

Agent Training

Training Setup

The environment and the RL agent are initialized first. The training setup includes:

- **Reward Shaping**: Modifying the reward structure to provide more meaningful feedback to the agent, encouraging desired behaviors.
- **Curriculum Learning**: Gradually increasing the difficulty of the task, such as making the terrain more challenging as the agent improves.

Training Process

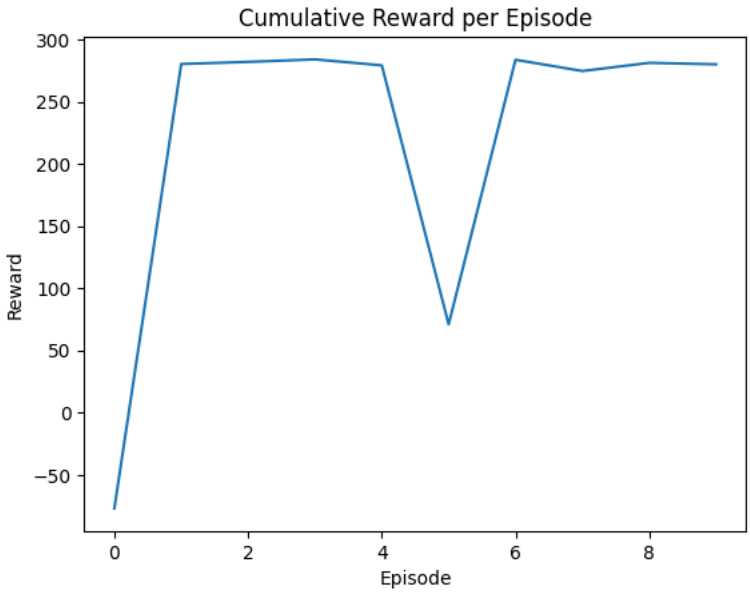
The agent is trained over 1000000 episodes to ensure it learns effectively. Key aspects of the training process include:

- **Experience Replay**: Storing and reusing past experiences to stabilize learning.
- **Action Noise**: Introducing noise to the actions to encourage exploration and prevent the agent from prematurely converging to suboptimal policies.

Performance Assessment

After training, the agent's performance is evaluated by running multiple episodes and recording the rewards. This helps in understanding how well the agent has learned to walk.

- **Visualization**: The training progress is visualized by plotting the cumulative reward per episode.



- **Video Recording**: A video of the agent performing well is recorded to provide a visual demonstration of its capabilities. (I uploaded the Video!)

- **Initial Performance**: The agent starts with a relatively low reward in the first episode. This is typical as the agent might still be exploring and adjusting its policy.

- **Consistency**: The rewards for most episodes (2, 3, 5, 6, 7, 8, and 9) are relatively high and stable, suggesting that the agent has learned to navigate the environment effectively. However, there is a notable dip in episode 5 where the reward drops significantly.

- **Fluctuations**: The sharp decline in episode 5 indicates that the agent may have encountered a scenario it couldn't handle well, leading to poor performance. This suggests variability in performance, possibly due to the stochastic nature of the environment or the agent's policy.

- **Recovery**: The agent recovers well after the dip, returning to high reward levels, which shows resilience and an ability to adapt.

Potential Areas for Improvement

1) Robustness: To prevent large drops in performance (like in episode 5), further training with a more diverse set of scenarios can help the agent become more robust. Techniques such as domain randomization can introduce variability during training, helping the agent generalize better.

2) Exploration vs. Exploitation: Introducing strategies to balance exploration and exploitation could prevent the agent from converging prematurely on suboptimal policies. Using techniques like epsilon-greedy or entropy regularization can help maintain exploration.

3) Reward Shaping: Refining the reward function to provide more granular feedback could help the agent learn more nuanced behaviors. For instance, rewarding partial achievements or penalizing specific undesirable behaviors (like stumbling) might help.

4) Experience Replay: Implementing a more sophisticated experience replay mechanism, such as prioritized experience replay, could improve learning efficiency by focusing on more informative experiences.

5) Hyperparameter Tuning: Further tuning of the hyperparameters (e.g., learning rate, batch size, discount factor) might yield better performance. Conducting systematic hyperparameter optimization can identify more effective configurations.

6) Policy Architecture: Experimenting with different neural network architectures (e.g., deeper networks, recurrent networks) might improve the agent's ability to learn and generalize complex behaviors.

7) Curriculum Learning: Gradually increasing the difficulty of the tasks during training (curriculum learning) can help the agent adapt to more challenging scenarios over time, leading to more stable performance across episodes.