Di-Gait-Tron
Reinforcement Learning Techniques for Bipedal Walker Training: A DDPG and DQN Approach

Biswajit Rana
B2330026 , M.Sc. BDA

Debayan Datta
B2330027 , M.Sc. BDA

INTRODUCTION

The bipedal walker is a classic reinforcement learning problem that simulates a two-legged robot navigating a challenging terrain. Training such a walker effectively requires sophisticated algorithms capable of learning optimal policies through trial and error. In this study, we explore two prominent reinforcement learning algorithms: Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG).

DQN, an extension of Q-learning that leverages deep neural networks, is particularly effective in discrete action spaces. It approximates the optimal action-value function, allowing the walker to make decisions based on its current state while maximizing cumulative rewards.

On the other hand, DDPG is designed for continuous action spaces, making it suitable for problems where actions involve fine motor control, such as adjusting joint angles in a bipedal walker.

The integration of these algorithms within the OpenAI Gym environment enables a robust framework for evaluating the performance and learning capabilities of the bipedal walker. By comparing the results from DQN and DDPG, we aim to identify the strengths and weaknesses of each approach in achieving stable and efficient locomotion.

This experiment contributes to the broader field of robotics and reinforcement learning, providing insights into the training processes necessary for developing autonomous agents capable of navigating complex environments.

RELATED WORKS

- **OpenAI's Baselines for Bipedal Walker**
  - Developed standard implementations of various RL algorithms, including DDPG and PPO.
  - Provided benchmarks for training bipedal walkers in the OpenAI Gym environment.
  - Focused on improving sample efficiency and stability of the training process.

- **Haarnoja et al. (2018) - Soft Actor-Critic (SAC)**
  - Proposed the Soft Actor-Critic algorithm, combining off-policy learning with entropy maximization.
  - Achieved stable locomotion in a bipedal walker by encouraging exploration through entropy.
  - Demonstrated improved performance in continuous action spaces compared to traditional methods.
- **Lillicrap et al. (2015) - DDPG Algorithm**
  - Introduced the Deep Deterministic Policy Gradient algorithm for continuous action spaces.
  - Applied DDPG to the bipedal walker problem, showcasing effective training and stability.
  - Emphasized the use of experience replay and target networks for enhanced learning efficiency.
- **Yarats et al. (2020) - A Study of DDPG and Soft Actor-Critic on Bipedal Walker**
  - Compared the performance of DDPG and Soft Actor-Critic across various locomotion tasks.
  - Evaluated algorithms based on sample efficiency and robustness during training.
  - Provided insights into the strengths and limitations of each approach in continuous control tasks.

PROPOSED MODEL

Deep Q-Network (DQN)

The Deep Q-Network (DQN) algorithm is a reinforcement learning technique that extends the traditional Q-learning algorithm by incorporating deep neural networks to approximate the Q-value function. This allows DQN to handle high-dimensional state spaces, such as those encountered in games and robotics.

- **Overview**:
  - DQN utilizes a neural network, referred to as the Q-network, to estimate the action-value function $Q(s, a)$, which represents the expected return of taking action a in state s.
  - The primary objective of DQN is to learn an optimal policy that maximizes the cumulative reward.

- **Algorithm Steps**:
  - **Initialization**: Initialize the Q-network $Q(s, a|\theta)$ with random weights θ. Also, initialize a target network $Q'(s, a|\theta')$ with the same weights.
  - **Experience Replay**: Maintain a replay buffer $\mathcal{D}$ to store transitions of the form $(s_t, a_t, r_t, s_{t+1}, d_t)$, where d_t indicates whether the episode has terminated.
  - **Action Selection**: For a given state s, select an action a using an ϵ-greedy strategy:

$$a = \{ \ random\, action\, with\, probability\ \epsilon$$

$$\{ \arg\max_a Q(s, a|\theta)\, with\, probability\ \ (1 - \epsilon)$$

  - **Store Transition**: Execute action a in the environment, observe reward r and next state s', and store the transition (s, a, r, s', d) in the replay buffer.
  - **Training**:
    - Sample a mini-batch of transitions from $\mathcal{D}$.
    - Calculate the target y for each sampled transition:

$$y = \{ r\ if\ dis\, True, r + \gamma \max_{a'} Q'(s', a'|\theta') if\ dis\, False$$

    - Perform a gradient descent step to minimize the loss:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} (y_i - Q(s_i, a_i|\theta))^2$$

  - **Update Target Network**: Periodically update the target network weights θ' by soft updating them with the Q-network weights:

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$$

    where τ is a small constant (e.g., $\tau = 0.001$).
- **Conclusion**:
  - DQN combines the benefits of deep learning and Q-learning, enabling it to successfully learn optimal policies in complex environments.
  - The use of experience replay and target networks significantly enhances the stability and convergence of the learning process.

Deep Deterministic Policy Gradient (DDPG)

The Deep Deterministic Policy Gradient (DDPG) algorithm is an off-policy reinforcement learning method that combines the strengths of deep learning and actor-critic methods. DDPG is particularly suitable for continuous action spaces and operates using two main components: an actor network and a critic network.

- **Overview**
  - The actor network selects actions based on the current state.
  - The critic network evaluates the actions taken by the actor by estimating the action-value function, $Q(s, a)$.
  - The primary objective of DDPG is to learn both the policy (actor) and the value function (critic) simultaneously.
- **Algorithm Steps**
  - **Initialization:**
    - Initialize the actor network $\mu(s|\theta^\mu)$ and the critic network $Q(s, a|\theta^Q)$ with random weights.
    - Initialize the target networks μ' and Q' with the same weights as the original networks.
  - **Experience Replay:**
    - Maintain a replay buffer $\mathcal{D}$ that stores transitions of the form (s_t, a_t, r_t, s_{t+1}).
  - **Action Selection:**
    - For a given state s, select an action $a = \mu(s|\theta^\mu) + \mathcal{N}$, where $\mathcal{N}$ is noise added for exploration.
  - **Store Transition:**
    - Execute action a in the environment, observe reward r and next state s', and store the transition (s, a, r, s') in the replay buffer.
  - **Training:**
    - Sample a mini-batch of transitions from $\mathcal{D}$.
    - Update the critic by minimizing the loss:

$$L = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - Q(s_i, a_i|\theta^Q)\right)^2$$

      where $y_i = r_i + \gamma Q'(s', \mu'(s'|\theta^{\mu'})|\theta^Q)$.
    - Update the actor using the policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_a Q(s, a|\theta^Q)\big|_{a=\mu(s|\theta^\mu)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)$$

  - **Soft Updates:**
    - Update the target networks using:

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$$

      where τ is a small value (e.g., $\tau = 0.001$).
  - **Repeat:**
    - Iterate through steps 3 to 6 until convergence or for a predefined number of episodes.
- **Conclusion**
  - DDPG leverages deep neural networks to approximate the actor and critic functions, enabling effective learning in continuous action spaces.
  - The combination of off-policy learning and experience replay makes DDPG a powerful algorithm for various control tasks in reinforcement learning.