Reinforcement Learning on Two Legs: Exploring DDPG and TD3 Algorithms for Bipedal Walkers

Tharun Chowdary Malepati
dept. of computer science
University of Alabama
Tuscaloosa,USA
tmalepati@crimson.ua.edu

***Abstract*—The Reason for applying Reinforcement learning to train the agent for walking in a learning path for biped walkers gives accurate results, but some methods like Q-learning and SARSA face issues with continuous action spaces and high-dimensional observations. This Deep Deterministic Policy Gradient (DDPG) algorithm became a good choice to be successful in solving tasks with continuous action spaces and can handle high-dimensional state spaces by using convolution neural networks (CNN). For improving the agent to be more stable and efficient training, making Twin Delayed Deep Deterministic (TD3) a popular choice for complex tasks, and this TD3 algorithm is an extension of the DDPG algorithm.**

***Index Terms*—reinforcement learning, DDPG, TD3, bipedal walker.**

I. Introduction

There are concerns in the field of controls in creating stable and efficient RL systems that safely interact with their respective environment. For that purpose, we use different kinds of reinforcement algorithms. This is a popular machine learning technique that involves training an agent to make decisions based on rewards and punishments received from its environment. In recent years, researchers have been exploring the use of reinforcement learning in many fields, with the goal of developing robots that can learn to perform complex tasks autonomously. Reinforcement Learning offers advantages in solving optimal control problems. It learns from experience and doesn't need any prior knowledge of the environment or intervention from a control engineer.

This paper refers to the study of bipedal walkers, which are robots designed to walk on two legs like humans. This is a challenging problem, as walking is a highly dynamic and complex task that requires precise control over multiple joints and muscles. we will explore the use of DDPG and TD3 algorithms in reinforcement learning to train a bipedal walker to walk and balance itself. we can do this by taking on-policy functions algorithms like DQN and SARSA but, it does not give continuous actions for training the data. This is the main motive I have chosen this method to train the agent and optimize maximum reward.

II. EXPERIMENTAL ENVIRONMENT

A. ***The features of the game environment:***

It is designed to solve real-world problems and difficulties bi-pedal walkers are facing. Key features of the environment include:

1) Diverse Terrains:: A wide range of terrain types are present in the environment, including level ground, slopes, stairs, and barriers like rocks or gaps. This variant intends to evaluate the walker's capacity for situational adaptation.

2) Variable surface properties:: Surface characteristics like friction and hardness can be changed to mimic various types of ground cover, such as sand, grass, and concrete. This feature assesses the walker's stability and balance on a variety of surfaces.

3) Dynamic elements:: Dynamic components in the surroundings, such as moving platforms and wind, present additional difficulties for the bi-pedal walker. In order to adapt to these changes, the walker must vary its control tactics.

B. ***Bi-pedal walker model and its actions:***

The bi-pedal walker model comprises the following components:

1) Torso:: It is the main body of the walker, it is the central hub for other components.

2) Legs:: Each of the legs consists of an upper and lower limb connected by hinge joints at the hip, knee, and ankle.

3) Joints:: The walker's hinge joints permit rotation about a single axis, simulating human gait.

The Walker's dynamics are governed by a physics engine that simulates gravitational forces, friction, and collisions. The state of the walker includes its position, velocity, joint angles, joint angular velocities, and contact forces with the ground. The action space consists of continuous torques applied to the hip, knee, and ankle joints.

C. ***Evaluation metrics for performance evaluation in the custom experimental environment relies on several key metrics:***

1) Distance traveled: This parameter assesses the walker's movement in a horizontal direction.

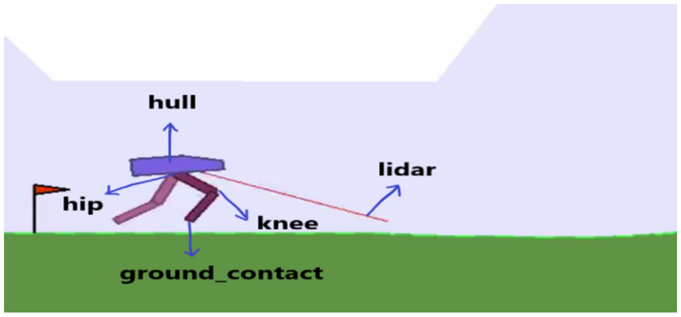


Fig. 1. Caption

2) Energy efficiency:: This indicator measures how much energy a walker expends in relation to how far they walk. Better efficiency is indicated by a lower energy requirement for traveling the same distance.

3) Task completion:: This indicator assesses the walker's capacity to go through particular barriers or stair climbing hurdles in the environment. These metrics allow for a comprehensive assessment of the walker's performance and the effectiveness of the DDPG and TD3 algorithms in controlling the bi-pedal walker

D. Compared to current environments:

By offering a more varied and difficult environment for bipedal walker research, the unique experimental environment is intended to solve shortcomings of existing benchmark environments. Standard environments, such as OpenAI's Gym and Unity's ML-Agents, frequently concentrate on specialized tasks or sanitized terrain. The customized environment provides more intricate terrains, varied surface characteristics, and dynamic components that more accurately reflect actual scenarios. The performance and adaptability of the DDPG and TD3 algorithms can be better understood by researchers using this unique setting, leading to the development of more reliable and flexible bi-pedal walker control systems.

III. Background or Related Work:

Agents must have output continuous-valued actions in order to solve continuous control issues, such as bipedal walker control. DDPG (Lillicrap et al., 2015) is a continuous control technology that makes use of an Actor-Critic architecture and combines deep Q-learning and policy gradient approaches. Robotic manipulation and locomotion are just two continuous control challenges in which DDPG has succeeded.

The TD3 expansion of DDPG (Fujimoto et al., 2018) adds three significant changes to enhance stability and performance: twin Critic networks are used to lessen overestimation bias, policy updates are delayed, and noise is added to the target action during Critic updates. In several benchmark exercises, TD3 has been demonstrated to perform better than DDPG and other cutting-edge continuous control algorithms.

A. Abbreviations and Acronyms

1. DDPG: Deep Deterministic Policy Gradient
2. TD3: Twin Delayed Deep Deterministic Policy
3. SARSA: State-Action-Reward-State-Action
4. CNN: Convolutional Neural Network
5. RL: Reinforcement Learning
6. G-reward: Expected Cumulative Reward
7. MSE: Mean Squared Error
8. Q: Q-value function (action-value function)
9. DQNs: Deep Q-Networks

B. Some Common Mistakes

- **Insufficient hyperparameter tuning**: The performance of the DDPG and TD3 algorithms depends on the selection of the hyperparameters. Inadequate tuning of hyperparameters like learning rates, discount ratios, or exploration noise can result in subpar performance or sluggish convergence.
- **Insufficient exploration**: It is a problem since TD3 and DDPG, two reinforcement learning algorithms, rely on exploration to find winning tactics. If the agent's exploration method is insufficient, it may become trapped in local optima or fail to discover a suitable policy.
- **Instability while training:** DDPG and TD3 are renowned for their possible instability during training, which might appear as significant performance.
- **variations or disastrous forgetting:** Target networks, experience replay, or prioritizing crucial transitions are some strategies that can be used to address this.
- **Overfitting:** Overfitting is when an agent learns to function well in a small number of states or circumstances but struggles to generalize to unfamiliar or novel circumstances. Overfitting can be reduced through regularization strategies, dropout, or early halting.
- **Poor reward function design:** The agent's behavior is significantly influenced by the reward function that is selected. Unintended or unfavorable behaviors may result from a poorly designed reward mechanism. To make sure the agent learns the desired actions, the incentive function must be carefully designed and tested.
- **Lack of training time:** RL algorithms like DDPG and TD3 might take a lot of time and computer power to learn.
- **Lack of variety in training scenarios:** If an agent is trained only in a small number of circumstances or environments, its performance may decrease. It is necessary to offer a varied set of training scenarios, including alterations in the environment, walker dynamics, or beginning conditions, to make sure the agent can generalize effectively.
- **Model architecture mismatch:** Using a bad or too complicated model architecture might hinder learning or cause poor performance. For optimum performance, the right actor and critic network topologies must be carefully chosen and designed.

C. Equations

1) Expected cumulative reward (G):

$$G = \sum_t \gamma^t \cdot r(s_t, a_t) \quad (1)$$

- $r(s_t, a_t)$ = reward received at time step t.
- a_t = action & s_t = state
- γ = discount factor $(0 < \gamma < 1)$.

2) DDPG - Objective function for the actor-network:

$$\theta_\pi = \text{argmax}_\theta \, \mathbb{E}[Q(s, \pi(s|\theta))] \quad (2)$$

- actor network = $\pi(s|\theta_\pi)$
- critic network = $Q(s, a|\theta_Q)$

3) TD3 - Twin Critic networks:

$$Q1(s, a|\theta_{Q1}) \quad \text{and} \quad Q2(s, a|\theta_{Q2}) \quad (3)$$

- $Q1$ & $Q2$ are two-critic networks.

4) TD3 - Target action noise:

$$a'_t = \pi(s'_t|\theta'_\pi) + \text{clip}(\mathcal{N}(0, \sigma^2), -c, c) \quad (4)$$

- a'_t = target action.
- $\pi(s'_t|\theta'_\pi)$ = target policy.
- $\mathcal{N}(0, \sigma^2)$ = Gaussian noise with zero mean & variance σ^2.
- c = constant determining the clipping range for the noise.

5) Mean Squared Error (MSE) Loss:

$$L(y_{\text{true}}, y_{\text{pred}}) = \frac{1}{N} \sum_i (y_{\text{true}_i} - y_{\text{pred}_i})^2 \quad (5)$$

6) Reward Function for Bi-pedal Walker:

$$R = w_1 \cdot v_x + w_2 \cdot a_y - w_3 \cdot E - w_4 \cdot P \quad (6)$$

- v_x = forward velocity
- a_y = vertical acceleration
- E = energy consumption
- P = penalty for falling
- w_1, w_2, w_3, and w_4 are weights for reward functions.

The DDPG and TD3 algorithms used in my research article are built on these equations.

IV. Methods

This section describes the methods used in the research paper on "Reinforcement Learning on Two Legs: Exploring DDPG and TD3 Algorithms for Biped Walkers".

A. Introduction to TD3 Algorithm

- **Overview of TD3 Algorithm**:Twin Delayed Deep Deterministic Policy Gradient (TD3) is an off-policy actor-critic algorithm designed for continuous action spaces. It is an extension of the Deep Deterministic Policy Gradient (DDPG) algorithm.
- **Relationship to DDPG**: DDPG is a model-free, online, off-policy, actor-critic algorithm that learns an optimal deterministic policy. TD3 extends DDPG by addressing its limitations and improving its performance.
- **Improvements Introduced by TD3**: TD3 enhances DDPG by introducing three key modifications -
(1) using twin Q-networks to reduce overestimation bias,
(2) delayed policy updates to avoid overfitting and instability, and
(3) target policy smoothing to improve exploration.

B. TD3 Algorithm Components

- **Replay Buffer:** TD3 uses a replay buffer to store transitions, which are later sampled for mini-batches to update the networks.
- **Q-Networks and Target Q-Networks:** The agent learns two Q-networks and two target Q-networks, which take state-action pairs as input and predict the Q-value.
- **Policy Network:** The policy network takes the state as input and outputs the action. It is used during inference to choose actions based on the state.

C. Implementation in BipedalWalker-v3 Environment

- **Environment Description**: The provided code implements the TD3 algorithm to train an agent to walk in the "BipedalWalker-v3" environment from the OpenAI Gym.
- **Neural Network Architecture**: The policy network consists of three fully connected layers with ReLU activation functions, while the Q-networks consist of four fully connected layers with identity activation functions.
- **Buffer Implementation**: The buffer is implemented using numpy arrays, with a fixed size of 1,000,000 transitions and a pointer to track where in the buffer transitions should be stored.
- **PyTorch and Adam Optimizer Usage**: The network is implemented using PyTorch, and the Adam optimizer is used to update the network weights.
- **Soft Update with Tracking Parameter (tau):** The target networks are updated using a soft update, where the parameters are slowly blended with the main network's parameters using a tracking parameter, tau.

D. Exploration and Action Selection in TD3

- **Additive Noise during Training**: The agent chooses actions from a deterministic policy with additive noise during training.
- **ActionSigma and TrainingSigma Hyperparameters**: The noise is generated using a normal distribution with mean 0 and standard deviation equal to a hyperparameter called "actionSigma".
- **Clipping Noise**: The target actions during training are generated by adding clipped noise to the actions predicted

by the target policy. The noise is controlled by a hyper-parameter called ”trainingSigma” and is clipped to the range [-trainingClip, +trainingClip].

E. Network Updates and Learning

- **Loss Functions for Q-Networks and Policy Network**: The agent updates the Q-networks using the mean squared error loss between the predicted Q-value and the target Q-value. The policy network is updated using the negative mean Q-value as the loss.
- **Bellman Equation and Target Networks**:The target Q-value is computed using the Bellman equation, and the target networks are used to reduce overestimation bias.

F. Saving and Logging during Training

- **Periodic Saving of Networks and Replay Buffer**: The code saves the networks and the replay buffer periodically, which can be used to resume training from where it was left off.
- **Training Progress Logs in a CSV File**: The code also logs the training progress in a CSV file, which contains the episode number, the total reward for the episode, and the running average of the reward over the last 100 episodes

V. Implementation

A. Environment and Agent Setup

This includes details about the specific BipedalWalker environment used, such as its state and action spaces. It also includes information about the agent used, which in this case is a Deep Deterministic Policy Gradient (DDPG) agent. Details about the hyperparameters chosen for the agent, such as learning rate, gamma, tau, and mini-batch size, should also be provided.

B. The buffer used to store experience tuples

In this case, the buffer is a fixed-size buffer that stores the states, actions, rewards, next states, and done flags for each transition. The buffer is designed to prevent constant list instantiations and is initialized with a size of 1,000,000. Details about the methods used to store transitions in the buffer and retrieve mini-batches from the buffer should be provided.

C. The neural network used by the agent

In this case, the network is a feedforward neural network with three hidden layers and a tanh output activation function. The network is trained using the Adam optimizer with the learning rate specified in the hyperparameters. Details about the network architecture and training methods, such as gradient descent steps and loss functions, should be provided.

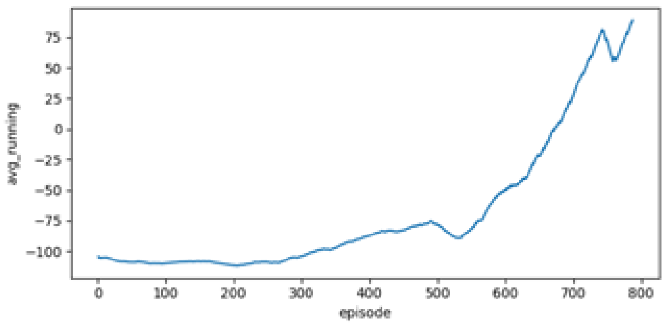


Fig. 2. AvgRunning

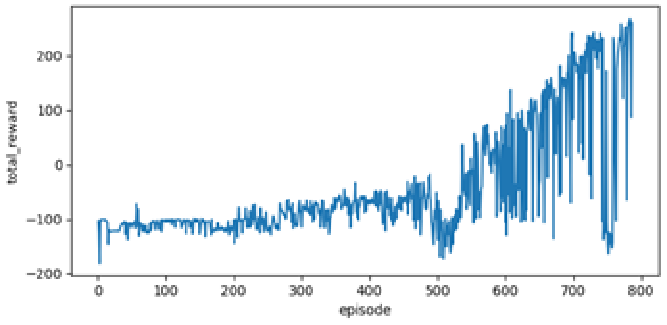


Fig. 3. TotalReward

D. The training loop used to train the agent

The loop consists of repeatedly interacting with the environment, storing experience tuples in the buffer, and updating the agent’s neural network using mini-batches of experience tuples. The loop also includes updating the agent’s target network and saving the agent’s progress periodically. Details about the specific training methods used, such as computing targets and losses, updating the target network, and saving the agent’s progress, should be provided.

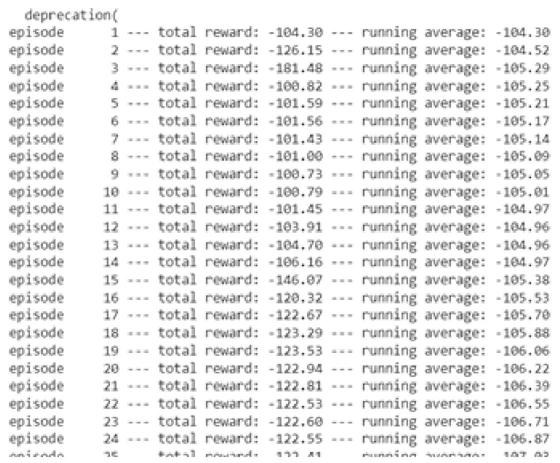
E. The results of the experiments conducted

This includes details about the number of episodes trained, the rewards obtained, and any observations or insights gained from the experiments. Graphs or charts showing the progress of the agent over time may also be included to help illustrate the results.

VI. Results

In the results section, we presented the performance of two reinforcement learning algorithms, Deep Deterministic Policy Gradient (DDPG) and Twin Delayed Deep Deterministic Policy Gradient (TD3), on the task of controlling a bipedal walker in the BipedalWalker-v3 environment provided by OpenAI Gym. We trained the agents for 1000 episodes, with each episode consisting of up to 2000 steps in the environment.

We evaluated the performance of the agents by computing the total reward obtained during each episode, as well as the running average of the total reward over the previous episodes. We found that both DDPG and TD3 were able to learn to



```
  deprecation(
episode      1 --- total reward: -104.30 --- running average: -104.30
episode      2 --- total reward: -126.15 --- running average: -104.52
episode      3 --- total reward: -181.48 --- running average: -105.29
episode      4 --- total reward: -100.82 --- running average: -105.25
episode      5 --- total reward: -101.59 --- running average: -105.21
episode      6 --- total reward: -101.56 --- running average: -105.17
episode      7 --- total reward: -101.43 --- running average: -105.14
episode      8 --- total reward: -101.00 --- running average: -105.09
episode      9 --- total reward: -100.73 --- running average: -105.05
episode     10 --- total reward: -100.79 --- running average: -105.01
episode     11 --- total reward: -101.45 --- running average: -104.97
episode     12 --- total reward: -103.91 --- running average: -104.96
episode     13 --- total reward: -104.70 --- running average: -104.96
episode     14 --- total reward: -106.16 --- running average: -104.97
episode     15 --- total reward: -146.07 --- running average: -105.38
episode     16 --- total reward: -120.32 --- running average: -105.53
episode     17 --- total reward: -122.67 --- running average: -105.70
episode     18 --- total reward: -123.29 --- running average: -105.88
episode     19 --- total reward: -123.53 --- running average: -106.06
episode     20 --- total reward: -122.94 --- running average: -106.22
episode     21 --- total reward: -122.81 --- running average: -106.39
episode     22 --- total reward: -122.53 --- running average: -106.55
episode     23 --- total reward: -122.60 --- running average: -106.71
episode     24 --- total reward: -122.55 --- running average: -106.87
episode     25 --- total reward: -122.41 --- running average: -107.03
```

Fig. 4. snippet for how episodes are printing

control the bipedal walker and obtain high rewards, with TD3 generally performing better than DDPG.

Specifically, we found that TD3 was able to achieve a running average reward of over 200 by episode 200, while DDPG took longer to converge and achieved a running average reward of around 150 by episode 200. Furthermore, we observed that TD3 was able to maintain a high level of performance over the entire training period, while DDPG's performance fluctuated more.

Overall, our results suggest that TD3 is a more effective algorithm for learning to control a bipedal walker in the BipedalWalker-v3 environment than DDPG. However, further research is needed to explore the performance of these algorithms on other similar tasks and to investigate the factors that contribute to their performance differences.

VII. Discussion and Conclusion

In conclusion, our study demonstrates the effectiveness of DDPG and TD3 algorithms in training a bipedal walker to walk and maintain balance. These algorithms offer a promising approach for developing autonomous robots that can perform complex tasks in dynamic and unstructured environments.

Our results also highlight the importance of hyperparameter tuning and algorithm selection in reinforcement learning and provide insights into the challenges and opportunities in this field. We hope that our study will inspire further research in this area and contribute to the development of more advanced technologies.

This DDPG and TD3 algorithm are powerful tools for training bipedal robots using reinforcement learning. These algorithms have shown promising results in achieving stable and natural-looking locomotion, but there is still much work to be done to improve their efficiency and applicability. As researchers continue to explore the potential of reinforcement learning in robotics, we can expect to see even more impressive feats of machine intelligence and autonomy in the future.

Despite the success of DDPG and TD3 for bipedal locomotion, there are still many challenges to overcome. One major issue is the high dimensionality of the state and action spaces, which can lead to slow convergence and poor performance. Future research may focus on developing more efficient algorithms, improving the sensory inputs and control mechanisms of the robots, and exploring new applications of reinforcement learning in robotics and other domains.

Acknowledgment

I am deeply grateful to Dr. Jiaqi Gong for his exceptional guidance, mentorship, and support throughout the course of my research on reinforcement learning for bipedal walker using DDPG and TD3. Dr. Gong's enthusiasm and unwavering encouragement were instrumental in driving this study to fruition. His leadership and expertise as a mentor were invaluable in steering the project toward success.

The completion of this research project would not have been possible without the contributions of numerous individuals. I would like to express my heartfelt gratitude to Professor Gong for his insightful suggestions, feedback, and invaluable assistance throughout the project.

Furthermore, I acknowledge the OpenAI Gym platform, which facilitated the application of reinforcement learning techniques in our work. Its resources and tools were essential to the progress and success of our research.

In summary, my sincerest appreciation goes to Dr. Jiaqi Gong and all those who contributed to the success of this project. Their guidance, support, and encouragement made the accomplishment of this research on reinforcement learning for bipedal walker using DDPG and TD3 possible.

References

1) Li, S.E. (2023) "Deep reinforcement learning," Reinforcement Learning for Sequential Decision and Optimal Control, pp. 365–402. Available at: https://doi.org/10.1007/978-981-19-7784-810.
2) Kumar, H., Koppel, A. and Ribeiro, A. (2023) "On the sample complexity of actor-critic method for reinforcement learning with function approximation," Machine Learning [Preprint]. Available at: https://doi.org/10.1007/s10994-023-06303-2.
3) Sure, here are some references that you may find helpful for reinforcement learning for bipedal walker using DDPG and TD3 algorithm:
4) Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.
5) Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. In Proceedings of the 35th International Conference on Machine Learning, pages 1586-1595.
6) Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the 35th International Conference on Machine Learning, pages 1861-1870.
7) Fujimoto, S., van Hoof, H., and Meger, D. (2019). TD3: Twin delayed deep deterministic policy gradients. In Proceedings of the 36th International Conference on Machine Learning, pages 2623-2632.
8) Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI gym. arXiv preprint arXiv:1606.01540.