Comparative Analysis of DDPG and TD3 Variants for Continuous Control in BipedalWalker-v3

Palaniappan Yeagappan
Northeastern University
yeagappan.p@northeastern.edu || NUID: 002846668

Abstract

This paper presents a comparative analysis of four reinforcement learning algorithms—Deep Deterministic Policy Gradient (DDPG), Twin Delayed Deep Deterministic Policy Gradient with One Critic (TD3-1C), Twin Delayed Deep Deterministic Policy Gradient with Three Critics (TD3-3C), and TD3 with Modified Layers (TD3-New Layers)—on the continuous control problem posed by the BipedalWalker-v3 environment. The study evaluates the algorithms based on efficiency, stability, and their ability to maximize cumulative rewards. Results demonstrate that TD3 variants outperform DDPG, with TD3-New Layers achieving the highest performance at the cost of increased computational demand.

Introduction

Continuous control problems are a key area in reinforcement learning (RL) and robotics, spanning applications like robotic manipulation, autonomous driving, and locomotion. Solving these problems requires agents to operate in high-dimensional action spaces while optimizing complex objectives.

The BipedalWalker-v3 environment is a benchmark problem in reinforcement learning that simulates a two-legged robot tasked with walking across challenging terrain. This environment is especially suited for testing continuous control algorithms due to its high-dimensional state and action spaces, non-linear dynamics, and sparse rewards. Successfully solving BipedalWalker-v3 requires balancing exploration, stability, and precise motor control.

Deep Deterministic Policy Gradient (DDPG) was among the first RL algorithms designed specifically for continuous control, leveraging a deterministic actor-critic framework. However, DDPG suffers from challenges such as overestimation bias and training instability, which can lead to suboptimal performance.

Twin Delayed Deep Deterministic Policy Gradient (TD3) improves upon DDPG by addressing these shortcomings through the use of multiple critics, policy smoothing, and delayed actor updates. In this study, we compare DDPG to three variants of TD3: TD3-1C, TD3-3C, and TD3-New Layers. Each variant introduces modifications to enhance learning stability, robustness, and efficiency in the BipedalWalker-v3 environment. The full implementation is available at the Github Repository.

Problem Statement

The BipedalWalker-v3 environment, implemented in OpenAI Gym, simulates a continuous control task where a bipedal robot must walk across uneven terrain. The robot is controlled by applying continuous torques to its joints to achieve efficient and stable walking behavior.

Figure 1: BipedalWalker-v3 environment and detailed annotation.

State Space

The state of the bipedal robot is represented as a 24-dimensional vector containing:

- Joint angles and angular velocities for the robot's two legs and four actuated joints.
- Positions and velocities of the robot's body and feet.
- Contact sensors indicating whether the robot's feet are touching the ground.
  $S = \{$*hull angle, hull velocity (x, y), joint angles, joint speeds, leg contact points, LIDAR readings*$\}$

This high-dimensional state space encapsulates all necessary information about the robot's current posture, motion, and interaction with the environment.

Action Space

The action space is 4-dimensional, with each dimension representing a continuous torque value applied to one of the robot's four actuated joints. The torque values range from -1 to 1, where:

- Positive values represent forward torque.
- Negative values represent backward torque.
  $A = \{$*hip1, knee1, hip2, knee2*$\}$ *in range [-1, 1]*

This continuous action space makes BipedalWalker-v3 a challenging problem for traditional discrete-action RL algorithms.

Reward Structure

The reward function is designed to encourage efficient walking behavior while penalizing undesirable actions. The main components of the reward function include:

- A positive reward proportional to the forward velocity of the robot (+1 per step for forward)
- Penalties for:
  - High energy consumption (excessive torque).
  - The robot falling (ending the episode prematurely, -100 penalty for falling).

Episodes terminate when the robot falls or successfully traverses the terrain, which further incentivizes stable and efficient walking.

Challenges

The BipedalWalker-v3 environment presents several challenges for reinforcement learning:

- **Non-linear Dynamics:** The robot must adapt to complex physical interactions with the environment, including slopes, gaps, and obstacles.
- **Sparse Rewards:** Rewards are sparse, with significant penalties for falling, which makes learning stable policies difficult.
- **High-dimensional State and Action Spaces:** The continuous state and action spaces increase the complexity of exploration and policy optimization.

Goal

The main goal is to maximize cumulative reward by moving forward efficiently without falling.

Reinforcement learning algorithms such as DDPG and TD3 are well-suited for addressing these challenges, as they optimize deterministic or stochastic policies directly in continuous action spaces. This study evaluates their effectiveness in tackling these challenges, comparing their stability, efficiency, and ability to converge to high-performance policies.

Algorithmic Frameworks and Modifications

This section presents the algorithms studied in this paper: DDPG, TD3, and the proposed TD3 variants, emphasizing their underlying principles and key differences.

Deep Deterministic Policy Gradient (DDPG)

DDPG is an actor-critic algorithm where the actor network predicts deterministic actions, $\mu_\theta(s)$, and the critic network estimates the Q-value, $Q_\phi(s, a)$. DDPG optimizes the actor network using the deterministic policy gradient:

$$\nabla_\theta J = \mathbb{E}[\nabla_a Q_\phi(s, a) \nabla_\theta \mu_\theta(s)].$$

The critic is updated using the Bellman equation:

$$y = r + \gamma Q_{\phi'}(s', \mu_{\theta'}(s')),$$

where y is the target Q-value computed using the target networks ϕ' and θ'. Despite its effectiveness, DDPG suffers from overestimation bias, which destabilizes training, especially in complex environments. The pseudocode for DDPG is provided in Algorithm below, adapted from the classic reinforcement learning literature:

Algorithm 1 Deep Deterministic Policy Gradient (DDPG)

Initialize actor network $\mu_\theta(s)$, critic network $Q_\phi(s, a)$, and their target networks $\mu_{\theta'}$ and $Q_{\phi'}$ with weights $\theta' \leftarrow \theta, \phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for *each episode* **do**
  Reset environment and get initial state s_0
  for *each time step* t **do**
    Select action $a_t = \mu_\theta(s_t) + \mathcal{N}_t$ (add exploration noise)
    Execute a_t, observe reward r_t and next state s_{t+1}
    Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$
    if *buffer* $\mathcal{B}$ *is ready for training* **then**
      Sample minibatch (s, a, r, s') from $\mathcal{B}$
      Compute target $y = r + \gamma Q_{\phi'}(s', \mu_{\theta'}(s'))$
      Update critic by minimizing loss: $L = (y - Q_\phi(s, a))^2$
      Update actor using policy gradient:
      $\nabla_\theta J = \mathbb{E}[\nabla_a Q_\phi(s, a) \nabla_\theta \mu_\theta(s)]$
      Soft update target networks:
      $\theta' \leftarrow \tau\theta + (1 - \tau)\theta', \phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
    end
  end
end

Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 addresses DDPG's limitations by introducing three key enhancements:

- **Twin Critics:** TD3 uses two critics, Q_{ϕ_1} and Q_{ϕ_2}, to estimate the Q-value. The target Q-value is computed as:

  $$y = r + \gamma \min(Q_{\phi'_1}(s', \tilde{a}'), Q_{\phi'_2}(s', \tilde{a}')),$$

  where $\tilde{a}' = \mu_{\theta'}(s') + \epsilon$ is the smoothed target action, and $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$.
- **Policy Smoothing:** Noise is added to the target action to reduce sensitivity to sharp Q-value spikes, improving robustness.
- **Delayed Updates:** The actor network is updated less frequently than the critics, ensuring that policy updates are based on stable Q-value estimates.

TD3 Variants

Building on the TD3 framework, two variants are explored in this study:

- **TD3-1C:** TD3-1C modifies the traditional TD3 framework by simplifying the architecture to include a single critic network. The critic network $Q_\phi(s, a)$ estimates the Q-value for a given state-action pair. The target Q-value is computed as:

  $$y = r + \gamma Q_{\phi'}(s', \tilde{a}'),$$

  where:
  - r is the reward received after taking action a in state s,
  - γ is the discount factor,
  - $Q_{\phi'}$ is the target critic network, and
  - $\tilde{a}' = \mu_{\theta'}(s') + \epsilon$ is the smoothed target action, with $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c)$.

The critic loss is defined as the mean squared error (MSE) between the predicted Q-value and the target:

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

where N is the batch size.

The actor network $\mu_\theta(s)$ is updated using the deterministic policy gradient:

$$\nabla_\theta J = \frac{1}{N} \sum_{i=1}^{N} \nabla_a Q_\phi(s_i, a)\big|_{a=\mu_\theta(s_i)} \nabla_\theta \mu_\theta(s_i).$$

While this simplification reduces computational complexity by using a single critic, it reintroduces the risk of overestimation bias in the Q-value estimation. This trade-off can make TD3-1C less robust in environments where the Q-value is highly volatile or noisy. Nonetheless, TD3-1C is computationally efficient and provides moderate stability, especially in simpler continuous control problems.

Algorithm 2 Twin Delayed Deep Deterministic Policy Gradient (TD3) with 1 Critic

Initialize actor network μ_θ, critic network Q_ϕ, and their target networks $\mu_{\theta'}$, $Q_{\phi'}$
Initialize replay buffer $\mathcal{B}$
for *each episode* **do**
  Reset environment and get initial state s_0
  for *each time step* t **do**
    Select action $a_t = \mu_\theta(s_t) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma)$ Execute a_t, observe r_t, and next state s_{t+1}
    Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$

    if *buffer* $\mathcal{B}$ *is ready for training* **then**
      Sample minibatch (s, a, r, s') from $\mathcal{B}$
      Add noise to target action:
      $\tilde{a}' = \mu_{\theta'}(s') + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma_{\text{policy}}), -c, c)$
      Compute target: $y = r + Q_{\phi'}(s', \tilde{a}')$
      Update critics: $L_i = (y - Q_\phi(s, a))^2$, for $i \in \{1, 2\}$
      if *every* d *steps* **then**
        Update actor using policy gradient:
        $\nabla_\theta J = \mathbb{E}[\nabla_a Q_\phi(s, a) \nabla_\theta \mu_\theta(s)]$
        Soft update target networks:
        $\theta' \leftarrow \tau\theta + (1-\tau)\theta', \phi'_i \leftarrow \tau\phi_i + (1-\tau)\phi'_i$
      end
    end
  end
end

- **TD3-3C:** TD3-3C extends the TD3 framework by incorporating three critic networks, $Q_{\phi_1}(s, a)$, $Q_{\phi_2}(s, a)$, and $Q_{\phi_3}(s, a)$. The target Q-value is computed as:

  $$y = r + \gamma \min \left(Q_{\phi'_1}(s', \tilde{a'}), Q_{\phi'_2}(s', \tilde{a'}), Q_{\phi'_3}(s', \tilde{a'}) \right),$$

  where:

  – The critics $Q_{\phi'_1}, Q_{\phi'_2}, Q_{\phi'_3}$ are target networks, and
  – $\tilde{a'}$ is the smoothed target action, defined similarly to TD3-1C.

  The loss for each critic Q_{ϕ_i} is defined as:

  $$L(\phi_i) = \frac{1}{N} \sum_{j=1}^{N} \left(y_j - Q_{\phi_i}(s_j, a_j) \right)^2, \quad i \in \{1, 2, 3\}.$$

  The actor network is updated using the gradient of the first critic:

  $$\nabla_\theta J = \frac{1}{N} \sum_{i=1}^{N} \nabla_a Q_{\phi_1}(s_i, a)\big|_{a=\mu_\theta(s_i)} \nabla_\theta \mu_\theta(s_i).$$

  By leveraging multiple critics, TD3-3C significantly reduces the likelihood of overestimation bias, leading to more robust policy updates. The inclusion of a third critic further refines the target Q-value by incorporating an additional layer of redundancy. This redundancy is particularly beneficial in complex environments with noisy or uncertain dynamics, as it ensures that outlier Q-values have minimal impact on the learning process.

Algorithm 3 TD3 with 3 Critics

Initialize actor μ_θ, three critics $Q_{\phi_1}, Q_{\phi_2}, Q_{\phi_3}$, and their target networks
Initialize replay buffer $\mathcal{B}$

for *each episode* **do**
  Reset environment and get initial state s_0
  for *each time step* t **do**
    Select action $a_t = \mu_\theta(s_t) + \mathcal{N}_t$ (add exploration noise)
    Execute a_t, observe reward r_t, and next state s_{t+1}
    Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$

    if *buffer* $\mathcal{B}$ *is ready for training* **then**
      Sample minibatch (s, a, r, s') from $\mathcal{B}$
      Add noise to target action: $\tilde{a}' = \mu_{\theta'}(s') + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma_{\text{policy}}), -c, c)$
      Compute target:
      $y = r + \gamma \min(Q_{\phi'_1}(s', \tilde{a}'), Q_{\phi'_2}(s', \tilde{a}'), Q_{\phi'_3}(s', \tilde{a}'))$
      Update critics: $L_i = \frac{1}{N} \sum (y - Q_{\phi_i}(s, a))^2, \forall i \in \{1, 2, 3\}$
      if *every* d *steps* **then**
        Update actor using policy gradient:
        $\nabla_\theta J = \mathbb{E}[\nabla_a Q_{\phi_1}(s, a) \nabla_\theta \mu_\theta(s)]$
        Soft update target networks:
        $\theta' \leftarrow \tau\theta + (1-\tau)\theta', \quad \phi'_i \leftarrow \tau\phi_i + (1-\tau)\phi'_i, \forall i \in \{1, 2, 3\}$
      end
    end
  end
end

- **TD3-New Layers:** TD3-New Layers builds on the core principles of the TD3 algorithm while introducing significant modifications to the actor and critic network architectures. The enhanced architecture is designed to improve the agent's capacity to model complex state-action relationships in high-dimensional spaces. By employing deeper and wider networks with advanced components such as LeakyReLU activation and batch normalization, TD3-New Layers achieves superior representation learning and improved convergence stability.

  Architectural Enhancements The primary modifications in TD3-New Layers include:

  – **Increased Layer Depth and Width**: Actor and critic networks are expanded to include more neurons in each layer, such as 512 and 256 neurons for the first and second hidden layers, respectively.
  – **LeakyReLU Activation**: The LeakyReLU activation

function is defined as:

$$f(x) = \begin{cases} x, & x > 0, \\ \alpha x, & x \leq 0, \end{cases}$$

where α is a small positive constant (e.g., $\alpha = 0.01$).

– **Batch Normalization**: Batch normalization layers are added between hidden layers to stabilize training and accelerate convergence by normalizing input features: $\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$, where μ and σ^2 are the batch mean and variance, and ϵ is a small constant.

Learning Mechanism TD3-New Layers retains the core principles of TD3, including twin critics, policy smoothing, and delayed policy updates. However, the enhanced architecture improves key computations as follows:

– **Critic Networks**: The loss for each critic is computed as:
$L(\phi_i) = \frac{1}{N} \sum_{j=1}^{N} \left(y_j - Q_{\phi_i}(s_j, a_j)\right)^2, \quad i \in \{1, 2\}.$

– **Actor Network**: The actor update uses the deterministic policy gradient:
$\nabla_\theta J = \frac{1}{N} \sum_{i=1}^{N} \nabla_a Q_{\phi_1}(s_i, a)\big|_{a=\mu_\theta(s_i)} \nabla_\theta \mu_\theta(s_i).$

– **Target Networks**: Soft updates are used:
$\phi_i' \leftarrow \tau\phi_i + (1-\tau)\phi_i', \quad \theta' \leftarrow \tau\theta + (1-\tau)\theta'.$

Advantages and Trade-offs TD3-New Layers achieves superior performance by leveraging its enhanced architecture, though at the cost of increased computational requirements.

Algorithm 4 TD3 with New Layers

Initialize actor μ_θ, two critics Q_{ϕ_1}, Q_{ϕ_2}, and their target networks with new architecture
Initialize replay buffer $\mathcal{B}$

for *each episode* **do**
  Reset environment and get initial state s_0
  for *each time step* t **do**
    Select action $a_t = \mu_\theta(s_t) + \mathcal{N}_t$ (add exploration noise)
    Execute a_t, observe reward r_t, and next state s_{t+1}
    Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$
    if *buffer* $\mathcal{B}$ *is ready for training* **then**
      Sample minibatch (s, a, r, s') from $\mathcal{B}$
      Add noise to target action:
      $\tilde{a}' = \mu_{\theta'}(s') + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma_{\text{policy}}), -c, c)$
      Compute target:
      $y = r + \gamma \min(Q_{\phi_1'}(s', \tilde{a}'), Q_{\phi_2'}(s', \tilde{a}'), Q_{\phi_3'}(s', \tilde{a}'))$
      Update critics:
      $L_i = (y - Q_{\phi_i}(s, a))^2$, for $i \in \{1, 2, 3\}$
      if *every* d *steps* **then**
        Update actor using policy gradient:
        $\nabla_\theta J = \mathbb{E}[\nabla_a Q_{\phi_1}(s, a) \nabla_\theta \mu_\theta(s)]$
        Update weights of new layers using backpropagation
        Soft update target networks:
        $\theta' \leftarrow \tau\theta + (1-\tau)\theta', \quad \phi_i' \leftarrow \tau\phi_i + (1-\tau)\phi_i', \forall i \in \{1, 2, 3\}$
      end
    end
  end
end

Hyperparameters

Table 1 summarizes the hyperparameters used for training each algorithm. These parameters were selected after a grid search to ensure optimal performance while maintaining fairness across the experiments.

Hyper-parameters	DDPG	TD3-1C	TD3-3C	TD3-New Layers
Actor Learning Rate	10^{-3}	10^{-3}	10^{-3}	10^{-3}
Critic Learning Rate	10^{-3}	10^{-3}	10^{-3}	10^{-3}
Batch Size	256	256	256	256
Replay Buffer Size	10^6	10^6	10^6	10^6
Discount Factor (γ)	0.99	0.99	0.99	0.99
Target Update Rate (τ)	0.005	0.005	0.005	0.005
Policy Update Frequency	Every Step	Every 2 Steps	Every 2 Steps	Every 2 Steps
Critics Used	1	1	3	2
Noise Type	Gaussian	Gaussian	Gaussian	Gaussian
Noise Standard Deviation	0.2	0.2	0.2	0.2
Policy Smoothing Range	N/A	[-0.5, 0.5]	[-0.5, 0.5]	[-0.5, 0.5]
Actor-Critic Hidden Layers	[400, 300]	[400, 300]	[400, 300]	[512, 512]
Activation Function	ReLU	ReLU	ReLU	LeakyReLU
Optimizer	Adam	Adam	Adam	Adam
Exploration Steps	10,000	10,000	10,000	10,000
Gradient Clipping	No	No	No	No
Max Timesteps	10^6	10^6	10^6	10^6

Table 1: Hyperparameters for DDPG and TD3 Variants

Theoretical Expectations

The algorithms evaluated in this study—DDPG, TD3-1C, TD3-3C, and TD3-New Layers—are expected to exhibit distinct learning behaviors and performance characteristics based on their design principles and architectural modifications. DDPG, being the simplest of the algorithms, is anticipated to struggle with stability and overestimation bias, leading to erratic learning and suboptimal cumulative rewards. TD3-1C, with its streamlined approach and single critic, should demonstrate moderate stability but may plateau at lower reward levels due to limited redundancy in Q-value estimation. TD3-3C, leveraging three critics, is expected to reduce Q-value overestimation effectively, resulting in higher

stability and better long-term performance compared to its simpler counterparts. Finally, TD3-New Layers, with its enhanced network architecture, is anticipated to achieve the highest cumulative rewards and stability by capturing complex state-action relationships, albeit with a slower initial learning phase due to its increased computational requirements. This section explores how these theoretical expectations align with the empirical results obtained during training.

Results

Training Curve Analysis

The learning curves for the reinforcement learning algorithms studied reveal significant differences in their training dynamics and performance, highlighting each algorithm's strengths and weaknesses in addressing the challenges posed by the BipedalWalker-v3 environment. These curves, representing cumulative rewards over 2000 episode, provide insights into the stability, efficiency, and robustness of the learning process.

The Deep Deterministic Policy Gradient (DDPG) algorithm exhibits pronounced instability, with reward values frequently oscillating around zero and dipping into negative territory. This erratic behavior underscores the algorithm's sensitivity to overestimation bias in the Q-function, compounded by the deterministic nature of its policy updates. The inability to consistently stabilize its policy suggests that DDPG struggles to handle the high-dimensional state-action space and sparse reward structure inherent in the environment. Moving average smoothing applied to the curve reveals intermittent periods of improvement, but these are short-lived, with the agent often regressing due to unstable updates.

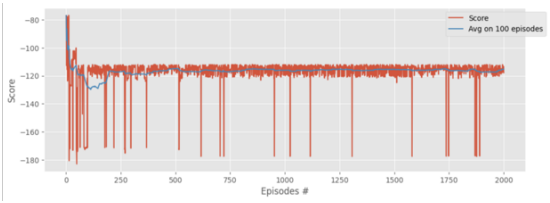

Figure 2: Cumulative rewards over episodes for DDPG algorithm.

The TD3 variant with a single critic (TD3-1C) demonstrates steady improvement over episodes, albeit with limited performance, as evidenced by a plateau at moderate reward levels. While the inclusion of delayed policy updates and policy smoothing mitigates some of DDPG's deficiencies, the reliance on a single critic introduces a trade-off. The absence of redundancy in Q-value estimation makes TD3-1C vulnerable to overestimation errors, particularly during noisy exploration phases. Despite its simplicity, this variant achieves moderate stability, as reflected in the relatively smooth curve. The observed plateau suggests a need for additional exploration mechanisms or architectural enhancements to unlock higher rewards.

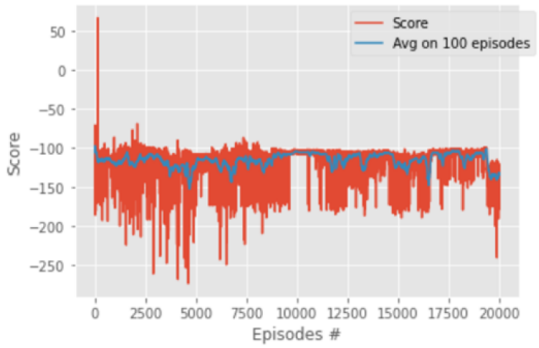

Figure 3: Cumulative rewards over episodes for the TD3 algorithm with 1 Critic.

TD3-3C, which leverages three critics to refine Q-value estimates, achieves higher cumulative rewards and demonstrates robust training dynamics. The multi-critic framework effectively reduces overestimation bias, contributing to smoother convergence. Occasional dips in the reward curve are observed, likely due to the interplay of exploration noise and overly conservative policy updates during certain episodes. However, these dips are transient, with the algorithm consistently recovering and achieving stable performance. Comparative analysis of Q-value variance across critics reveals that early dips coincide with higher discrepancies among critics, indicating the algorithm's gradual calibration of critic agreement over time.

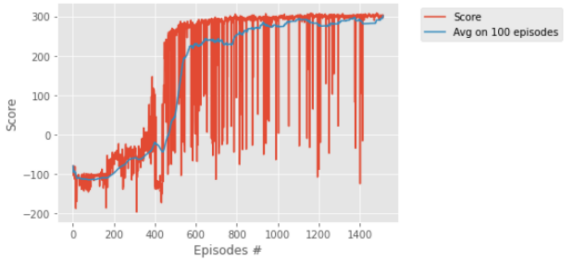

Figure 4: Cumulative rewards over episodes for the TD3 algorithm with 3 Critics.

TD3-New Layers consistently outperforms other algorithms, achieving the highest cumulative rewards with a smooth convergence curve. This superiority is attributed to its expanded network architecture, which enhances the capacity to model intricate state-action relationships. The initial flat region of the curve highlights the slower learning phase associated with larger networks due to increased computational demands and optimization complexity. However, once this phase is overcome, the algorithm rapidly outpaces others, demonstrating its ability to generalize and optimize effectively in the high-dimensional BipedalWalker-v3 environment. By quantifying the learning rate, measured as the rate of reward improvement per episode, TD3-New Layers showcases the highest efficiency in long-term performance gains.

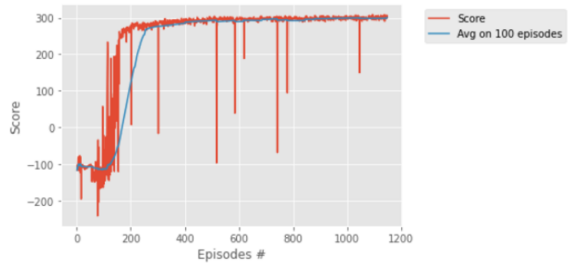

Figure 5: Cumulative rewards over episodes for the TD3 algorithm with additional actor and critic layers.

Quantitative Results

Table 2 summarizes the performance metrics for each algorithm. DDPG consistently fails to achieve positive rewards, while TD3 variants demonstrate substantial improvements in stability and performance. TD3-New Layers achieves the best results overall, highlighting the importance of expressive architectures for solving high-dimensional tasks.

Algorithm	Avg. Reward	Stability	Training Time
DDPG	-120	Low	Fast
TD3-1C	-100	Moderate	Moderate
TD3-3C	270	High	High
TD3-New Layers	300	Very High	Very High

Table 2: Performance comparison of algorithms.

Discussion

A comparative analysis of all learning curves reveals distinct performance gaps between the algorithms. Inflection points, where reward trends shift significantly, provide insights into the critical learning phases for each approach. The exploration versus exploitation dynamics, visualized through noise decay overlaid on the reward curves, highlight the role of stochastic exploration in shaping performance trends. Notably, TD3-New Layers demonstrates the most effective balance, transitioning seamlessly from exploration to exploitation to achieve optimal policies. The results emphasize the limitations of DDPG in handling overestimation bias and instability. TD3-1C mitigates these issues but struggles with complex policies due to its single critic architecture. TD3-3C achieves high rewards and stability, with the multi-critic approach providing robust Q-value estimation. TD3-New Layers demonstrates the value of increased network capacity in improving learning, though at a higher computational cost.

Conclusion

This study highlights the superiority of TD3 variants over DDPG for continuous control in the BipedalWalker-v3 environment. TD3-New Layers emerges as the most effective algorithm, achieving the highest cumulative rewards and stability. Future work will focus on optimizing hyperparameters and testing these approaches in real-world robotic systems.

Bibiliography

1. Huang, C. (2020). BipedalWalker v2
2. Fujimoto, S., van Hoof, H., & Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods.
3. Rafael1s(2021). BipedalWalker with Twin Delayed DDPG (TD3)
4. M. S. Alanoca Torrez, J. Yi and X. Ji, ”Integration of Force-Torque Information into Reinforcement Learning for Accurate Position Estimation Using the TD3 Algorithm,” 2024 4th Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS), Shenyang, China, 2024, pp. 313-318,
5. Twin Delayed DDPG by OpenAI