Dauphine | PSL
UNIVERSITÉ PARIS

MIDO
MATHÉMATIQUES ET INFORMATIQUE
DE LA DÉCISION ET DES ORGANISATIONS

M2 IASD

# Reinforcement Learning
# Topic: Bipedal Walker

Sepanta Farzollahi

*Instructor:* Stéphane Airiau

January 19, 2026

# Abstract

This report presents a reinforcement learning project focused on the control of a bipedal walker in the `BipedalWalker-v3` environment from Gymnasium. The main objective was to implement, train, and compare representative reinforcement learning algorithms to understand their behavior in a challenging continuous control task. Three actor–critic algorithms were selected for evaluation: the on-policy Proximal Policy Optimization (PPO), and the off-policy Soft Actor-Critic (SAC) and Twin Delayed Deep Deterministic Policy Gradient (TD3).

The report describes the environment, the theoretical foundations of each algorithm, the experimental setup, and the methods used to monitor and evaluate learning performance. Quantitative metrics such as episode reward and episode length are analyzed, and qualitative insights are provided through the observation of the agents' behavior during training and evaluation.

Limitations, hyperparameter choices, and potential extensions are discussed, highlighting avenues for further improvement and research in continuous control reinforcement learning.

# Contents

# Chapter 1

# Introduction

Reinforcement Learning (RL) addresses sequential decision-making problems in which an agent learns a policy through interaction with an environment. In continuous control settings, the agent must operate in high-dimensional state spaces and produce continuous-valued actions, while coping with delayed rewards and unstable training dynamics. These characteristics make continuous control tasks particularly challenging for deep reinforcement learning methods.

This project considers the *BipedalWalker* environment, a widely used benchmark for continuous control provided by the `Gymnasium` framework (Towers et al. (2025)) [1]. The task consists of controlling a simulated two-legged robot by applying continuous torque commands to its joints in order to move forward without falling. The environment exposes a continuous observation space describing the robot's physical state, and a continuous action space corresponding to joint motor commands. Due to its long-horizon dynamics and strict termination conditions, *BipedalWalker* represents a demanding test case for reinforcement learning algorithms.

The goal of this work is to analyze and compare the performance of three deep reinforcement learning algorithms on this environment: Proximal Policy Optimization (PPO) (Schulman et al. (2017)), Soft Actor-Critic (SAC) (Haarnoja et al. (2018)), and Twin Delayed Deep Deterministic Policy Gradient (TD3) (Fujimoto et al. (2018)). These methods were selected as representative actor-critic approaches for continuous control. PPO is an on-policy algorithm that enforces conservative policy updates to improve training stability, whereas SAC and TD3 are off-policy algorithms designed to achieve higher sample efficiency and robustness through experience replay and target networks.

All experiments were implemented using the `stable-baselines3` library (Raffin et al. (2021)) and rely on the `Gymnasium` API for environment interaction [2]. Each algorithm was trained with a fixed budget of 1.5 million environment steps under comparable conditions. Performance is evaluated using episode reward and episode length as primary metrics, complemented by qualitative analysis of the learned walking behaviors through environment rollouts.

This experimental study aims to provide empirical insights into the practical behavior of on-policy and off-policy reinforcement learning algorithms in a complex continuous locomotion task, and to highlight their respective strengths and limitations when applied to the *BipedalWalker* environment.

[1] The environment documentation: https://gymnasium.farama.org/environments/box2d/bipedal_walker/
[2] The `stable-baselines3` documentation: https://stable-baselines3.readthedocs.io/en/master/

# Chapter 2

# Environment Description

This chapter describes the *BipedalWalker* environment, a challenging benchmark for continuous control and reinforcement learning. It introduces the robot's observation and action spaces, as well as the reward structure and episode termination conditions. Understanding these elements is essential for designing and evaluating effective reinforcement learning algorithms in this environment.

## 2.1 The BipedalWalker Environment

The *BipedalWalker* environment is a continuous control task provided by the `Gymnasium` framework and built on top of the Box2D physics engine. It simulates a planar biped robot with four actuated joints that must learn to walk forward without falling. There are two versions of the environment: normal and hardcore. The normal version contains slightly uneven terrain, while the hardcore version includes more challenging obstacles such as ladders, stumps, and pitfalls. In this project, only the normal version of the environment is considered.

Each episode starts with the walker standing upright at the left end of the terrain with its hull horizontal and legs in a neutral configuration. The terrain is procedurally generated, requiring adaptive locomotion strategies. An episode ends either when the robot falls (e.g., the hull contacts the ground), when the walker reaches the right end of the terrain, or when the maximum number of timesteps is reached.

## 2.2 Observation Space

The observation space is a continuous 24-dimensional vector describing the dynamic state of the robot. It includes measurements such as joint angles and angular velocities, hull angle and angular velocity, horizontal and vertical velocities, leg contact indicators with the ground, and distance measurements obtained from lidar rangefinders. These observations capture both kinematic and dynamic information needed for control, but do not include explicit spatial coordinates of the walker in the environment.

Formally, the observation space can be represented as a box:

$$\text{observation\_space} = \text{Box}(-\textbf{high}, \textbf{high}, (24,), \texttt{float32})$$

where **high** is a vector containing the environment-specific upper bounds for each observation dimension, representing the maximum physically plausible values for the robot's state variables.

Each entry is bounded within $[-\mathbf{high}_i, \mathbf{high}_i]$, ensuring that all observations remain within reasonable and safe ranges for the learning agent.

## 2.3 Action Space

The action space is continuous and four-dimensional. Each element of the action vector corresponds to a motor speed command applied to one of the four joints (hips and knees). Actions are bounded within the range $[-1, +1]$, where extreme values represent maximum torque commands in either direction. This continuous formulation requires the use of policy gradient or actor-critic methods, as discrete action approaches such as standard Q-learning (Watkins and Dayan (1992)) are not directly applicable.

## 2.4 Reward Function

Rewards are structured to encourage forward progression and penalize inefficient or unstable control. At each timestep, the agent receives a positive reward proportional to its forward movement. There is also a small penalty applied for motor torque usage to discourage unnecessary energy consumption. Additionally, if the robot falls during an episode, it receives a large negative reward (e.g., $-100$) as a failure penalty. The cumulative reward over an episode can exceed 300 points for successful forward traversal of the environment.

This reward setup creates a multi-objective trade-off between achieving fast, stable locomotion and minimizing energy expenditure, which reinforces smooth and efficient walking behaviors.

## 2.5 Episode Termination

Episodes terminate under several distinct conditions:

- **Failure by falling**: If the main body (hull) of the walker contacts the ground, indicating a loss of balance.
- **Completion of the terrain**: When the walker successfully reaches the right end of the terrain before the timestep limit.
- **Time limit exceeded**: If the maximum timestep (e.g., 1600 for normal) is reached without failure or completion.

Episode length serves as a secondary performance indicator: longer episodes with high cumulative reward typically reflect more stable and competent locomotion policies.

# Chapter 3

# Reinforcement Learning Algorithms

This chapter presents the reinforcement learning algorithms evaluated in this project. All methods belong to the family of actor-critic algorithms and are designed for continuous control tasks. They differ mainly in their learning paradigm, which can be categorized into on-policy and off-policy approaches.

## 3.1 On-Policy Methods

On-policy algorithms learn a policy using data generated exclusively by the current policy. While this constraint can reduce sample efficiency, it often leads to more stable training dynamics, which is particularly important in high-dimensional continuous control tasks.

### 3.1.1 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is an on-policy actor-critic algorithm designed to achieve stable and reliable policy updates in reinforcement learning. The main challenge addressed by PPO is the instability that arises when policy parameters are updated too aggressively, which can lead to performance collapse, especially in high-dimensional continuous control tasks.

PPO optimizes a stochastic policy $\pi_\theta(a \mid s)$ parameterized by $\theta$, where $s_t$ denotes the state at time step $t$ and $a_t$ the action sampled from the policy. To measure how much the policy changes during training, PPO introduces the probability ratio:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)},$$

where $\pi_{\theta_{\text{old}}}$ is the policy used to collect the current batch of trajectories. This ratio quantifies how much more (or less) likely the new policy is to select the same action $a_t$ in state $s_t$ compared to the old policy. Values of $r_t(\theta)$ close to 1 indicate small policy updates, while large deviations indicate potentially unstable updates.

The core idea of PPO is to restrict policy updates by optimizing a clipped surrogate objective:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[ \min \left( r_t(\theta) A_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) A_t \right) \right],$$

where the expectation is taken over time steps sampled from the collected trajectories.

Each component of this objective has a specific role:

- $A_t$ is the advantage estimate at time step $t$, which measures how much better the chosen action $a_t$ performed compared to the average behavior of the policy in state $s_t$. A positive $A_t$ indicates that the action was better than expected, while a negative value indicates a suboptimal action.
- $\epsilon$ is a hyperparameter controlling the clipping range, typically set to a small value (e.g., $\epsilon = 0.2$). It defines how far the policy is allowed to deviate from the previous policy during a single update.
- The clip$(\cdot)$ operator limits the probability ratio $r_t(\theta)$ to the interval $[1 - \epsilon, 1 + \epsilon]$, preventing excessively large policy updates.

The use of the minimum operator ensures that the objective does not increase if the policy update moves outside the trusted region defined by the clipping range. As a result, PPO performs conservative updates that improve stability while still allowing meaningful policy improvement.

To compute the advantage estimates $A_t$, PPO relies on Generalized Advantage Estimation (GAE) (Schulman et al. (2016)). GAE computes advantages using a weighted sum of temporal-difference residuals:

$$A_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}, \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t),$$

where $\gamma$ is the discount factor, $\lambda$ controls the bias–variance trade-off, and $V(s)$ is the value function learned by the critic. This formulation reduces variance compared to Monte Carlo estimates while introducing only a limited bias.

In practice, PPO alternates between data collection and optimization phases. Trajectories are collected using the current policy, and multiple epochs of minibatch stochastic gradient descent are performed on the same batch of on-policy data. This allows efficient use of collected samples while maintaining the on-policy constraint.

PPO is particularly well-suited for the BipedalWalker environment due to its robustness to hyperparameter choices and its ability to maintain stable learning dynamics over long episodes and continuous action spaces. These properties make PPO a strong baseline for continuous locomotion tasks.

## 3.2 Off-Policy Methods

Off-policy algorithms learn a policy using data collected from potentially older versions of the policy, stored in a replay buffer. This allows for significantly improved sample efficiency, as each transition can be reused multiple times during training. However, off-policy learning introduces additional challenges related to stability and bias, which are addressed through target networks and delayed updates.

### 3.2.1 Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) is an off-policy actor-critic algorithm specifically designed for continuous control tasks. Its main distinguishing feature is the inclusion of an entropy maximization term in the learning objective, which encourages stochastic policies and sustained exploration

throughout training. This property improves robustness and helps prevent premature convergence to suboptimal deterministic behaviors.

SAC optimizes a stochastic policy $\pi_\theta(a \mid s)$ that outputs a probability distribution over actions rather than a single deterministic action. The learning objective is formulated in the maximum entropy reinforcement learning framework, where the policy seeks to maximize both the expected cumulative reward and the entropy of the action distribution:

$$J(\pi) = \mathbb{E}_\pi \left[ \sum_{t=0}^{T} \left( r(s_t, a_t) + \alpha \, \mathcal{H}(\pi(\cdot \mid s_t)) \right) \right],$$

where the expectation is taken over trajectories generated by the policy.

Each term in this objective plays a specific role:

- $r(s_t, a_t)$ is the reward received after executing action $a_t$ in state $s_t$, encouraging forward locomotion in the BipedalWalker environment.
- $\mathcal{H}(\pi(\cdot \mid s_t))$ denotes the entropy of the policy at state $s_t$, defined as

  $$\mathcal{H}(\pi(\cdot \mid s_t)) = -\mathbb{E}_{a \sim \pi} \left[ \log \pi(a \mid s_t) \right].$$

  Higher entropy corresponds to more stochastic action selection, promoting exploration.
- $\alpha$ is the temperature parameter that controls the relative importance of the entropy term. Large values of $\alpha$ encourage exploration, while smaller values prioritize reward maximization.

To estimate action values, SAC maintains two independent Q-functions, $Q_{\phi_1}(s, a)$ and $Q_{\phi_2}(s, a)$. During training, the minimum of the two estimates is used:

$$Q_{\min}(s, a) = \min \left( Q_{\phi_1}(s, a), Q_{\phi_2}(s, a) \right),$$

which reduces overestimation bias commonly observed in value-based methods.

The critic networks are trained using a soft Bellman backup:

$$y_t = r(s_t, a_t) + \gamma \, \mathbb{E}_{a' \sim \pi} \left[ Q_{\min}(s_{t+1}, a') - \alpha \log \pi(a' \mid s_{t+1}) \right],$$

where $\gamma$ is the discount factor. The entropy term inside the target encourages the critic to assign higher value to states where multiple good actions are available.

The policy parameters $\theta$ are updated by minimizing the following objective:

$$J_\pi(\theta) = \mathbb{E}_{s_t \sim \mathcal{D}, a_t \sim \pi_\theta} \left[ \alpha \log \pi_\theta(a_t \mid s_t) - Q_{\min}(s_t, a_t) \right],$$

where $\mathcal{D}$ denotes the replay buffer storing transitions collected during training. This formulation encourages the policy to select actions that both maximize expected return and maintain sufficient entropy.

In this project, the temperature parameter $\alpha$ is automatically tuned during training to match a target entropy value. This adaptive mechanism allows SAC to balance exploration and exploitation without manual tuning, which is particularly beneficial in complex continuous control environments.

SAC is well-suited for the BipedalWalker task due to its sample efficiency, stable off-policy learning, and ability to produce smooth and robust locomotion policies through stochastic exploration.

### 3.2.2 Twin Delayed Deep Deterministic Policy Gradient (TD3)

Twin Delayed Deep Deterministic Policy Gradient (TD3) is an off-policy actor-critic algorithm designed to address the instability and overestimation issues observed in the original Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al. (2016)). It is specifically tailored for continuous action spaces and environments requiring precise control, such as the BipedalWalker task.

TD3 introduces three main improvements over DDPG: clipped double Q-learning, delayed policy updates, and target policy smoothing.

**Clipped Double Q-Learning** TD3 maintains two independent critic networks, $Q_{\phi_1}(s, a)$ and $Q_{\phi_2}(s, a)$, each estimating the action-value function. During training, the minimum of the two Q-values is used to compute the target:

$$y_t = r(s_t, a_t) + \gamma \min_{i=1,2} Q_{\phi_i}(s_{t+1}, a'_{t+1}),$$

where:

- $r(s_t, a_t)$ is the reward received after executing action $a_t$ in state $s_t$.
- $\gamma$ is the discount factor, which balances immediate and future rewards.
- $a'_{t+1}$ is the action selected by the target policy (see below) at the next state $s_{t+1}$.

Taking the minimum of the two Q-values mitigates overestimation bias, a common source of instability in value-based actor-critic methods.

**Target Policy Smoothing** To further improve stability, TD3 adds noise to the target action when computing the critic target:

$$a'_{t+1} = \pi_{\theta'}(s_{t+1}) + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \sigma^2), -c, c),$$

where:

- $\pi_{\theta'}$ is the target actor network.
- $\mathcal{N}(0, \sigma^2)$ is Gaussian noise with standard deviation $\sigma$.
- The clipping interval $[-c, c]$ prevents large deviations in the target action.

This smoothing prevents the critic from overfitting to sharp peaks in the Q-function and encourages learning policies that are robust to small perturbations in actions.

**Delayed Policy Updates** Unlike the critic networks, the actor (policy) is updated less frequently, typically once every $d$ critic updates. This delay ensures that the policy is trained against more accurate Q-value estimates, reducing instability and oscillations during learning. Formally, the actor objective is:

$$J(\pi_\theta) = \mathbb{E}_{s \sim \mathcal{D}} \left[ -Q_{\phi_1}(s, \pi_\theta(s)) \right],$$

where $\mathcal{D}$ is the replay buffer of transitions collected during training, and $Q_{\phi_1}$ is the first critic network (the second critic is used only for the target). Minimizing this objective corresponds to maximizing the expected return under the current policy.

**Exploration Strategy** TD3 uses a deterministic policy but introduces exploration by adding noise to actions during environment interaction:

$$a_t = \pi_\theta(s_t) + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \sigma^2_{\text{explore}}),$$

where $\eta_t$ is Gaussian exploration noise. This approach allows the agent to explore the state-action space while ultimately learning a deterministic policy.

By combining clipped double Q-learning, target policy smoothing, and delayed actor updates, TD3 significantly improves stability and sample efficiency in continuous control tasks. In the BipedalWalker environment, TD3 learns smooth and stable locomotion policies, effectively balancing exploration and exploitation while mitigating common sources of instability observed in DDPG.

# Chapter 4

# Experimental Setup

This chapter describes the experimental configuration used to train and evaluate the reinforcement learning algorithms on the BipedalWalker environment. It details the implementation framework, training procedure, hyperparameters, and evaluation metrics in order to ensure reproducibility and fair comparison between algorithms.

## 4.1 Implementation Framework

All experiments were implemented in Python using the `Gymnasium` framework for environment interaction and the `stable-baselines3` library for reinforcement learning algorithms. Neural network policies and value functions rely on multilayer perceptrons provided by the default `MlpPolicy` architecture.

Training and logging were conducted using the built-in monitoring tools of `stable-baselines3`, which record episode rewards, episode lengths, and training timesteps. Learning curves were generated from these logs for subsequent analysis.

## 4.2 Environment Configuration

All experiments were performed on the `BipedalWalker-v3` environment using the normal version of the task. The observation space is a 24-dimensional continuous vector, and the action space is four-dimensional with continuous values bounded in the interval $[-1, 1]$.

To improve data efficiency and stabilize learning, vectorized environments were used when supported by the algorithm. PPO was trained using eight parallel environments, SAC using four parallel environments, and TD3 using a single environment, following standard practices for on-policy and off-policy methods.

## 4.3 Training Procedure

Each algorithm was trained for a total of 1.5 million environment interaction steps. For on-policy training with PPO, rollouts were collected using the current policy and discarded after each update. For off-policy methods (SAC and TD3), transitions were stored in a replay buffer and reused multiple times during training.

All experiments were conducted using a fixed random seed and identical environment configurations across algorithms in order to reduce variability and ensure fair comparison. No

additional reward shaping or curriculum learning strategies were applied beyond the default environment design.

## 4.4 Hyperparameters

Unless otherwise stated, default hyperparameters from `stable-baselines3` were used, with minor adjustments to improve convergence on the BipedalWalker task. The main hyperparameters for each algorithm are summarized below.

### 4.4.1 PPO Hyperparameters

PPO was trained with a learning rate of $3 \times 10^{-4}$ and a discount factor $\gamma = 0.99$. Rollouts of 2048 steps were collected per environment before each update. The clipped surrogate objective used a clipping parameter $\epsilon = 0.2$, and Generalized Advantage Estimation was applied with $\lambda = 0.95$.

Optimization was performed using minibatches of size 64 over 10 epochs per update. An entropy coefficient of 0.01 was included to encourage exploration during training.

### 4.4.2 SAC Hyperparameters

SAC was trained with a learning rate of $3 \times 10^{-4}$ and a discount factor $\gamma = 0.99$. A replay buffer of size $10^6$ transitions was used, with minibatches of size 256 sampled for each update.

The target smoothing coefficient was set to $\tau = 0.005$. The entropy temperature parameter $\alpha$ was automatically tuned during training, allowing the algorithm to adaptively control exploration without manual intervention.

### 4.4.3 TD3 Hyperparameters

TD3 used a learning rate of $3 \times 10^{-4}$ and a discount factor $\gamma = 0.99$. A replay buffer of size $10^6$ and a batch size of 256 were employed.

The target network update coefficient was set to $\tau = 0.005$, and the policy update delay was set to 2, meaning the actor was updated once every two critic updates. Exploration was encouraged through additive Gaussian noise applied to actions, with a standard deviation of 0.2. Target policy smoothing was implemented using clipped Gaussian noise with a standard deviation of 0.2 and a clipping range of 0.5.

## 4.5 Evaluation Metrics

Performance was evaluated using two primary metrics: episode reward and episode length. Episode reward measures the cumulative return obtained during an episode and directly reflects task performance. Episode length provides an additional indicator of stability, as longer episodes correspond to sustained walking without falling.

To reduce noise in the learning curves, a rolling mean with a window of 50 episodes was applied for visualization purposes. In addition to quantitative evaluation, qualitative assessment was performed by rendering trained policies and recording rollouts as animated sequences.

# Chapter 5

# Experimental Results and Analysis

This chapter presents and analyzes the experimental results obtained on the `BipedalWalker-v3` environment. The performance of PPO, SAC, and TD3 is evaluated using episode reward and episode length, both in raw form and smoothed using a rolling average. Quantitative results are complemented by qualitative visual inspection of trained policies.

## 5.1 Overview of the Comparison Figure

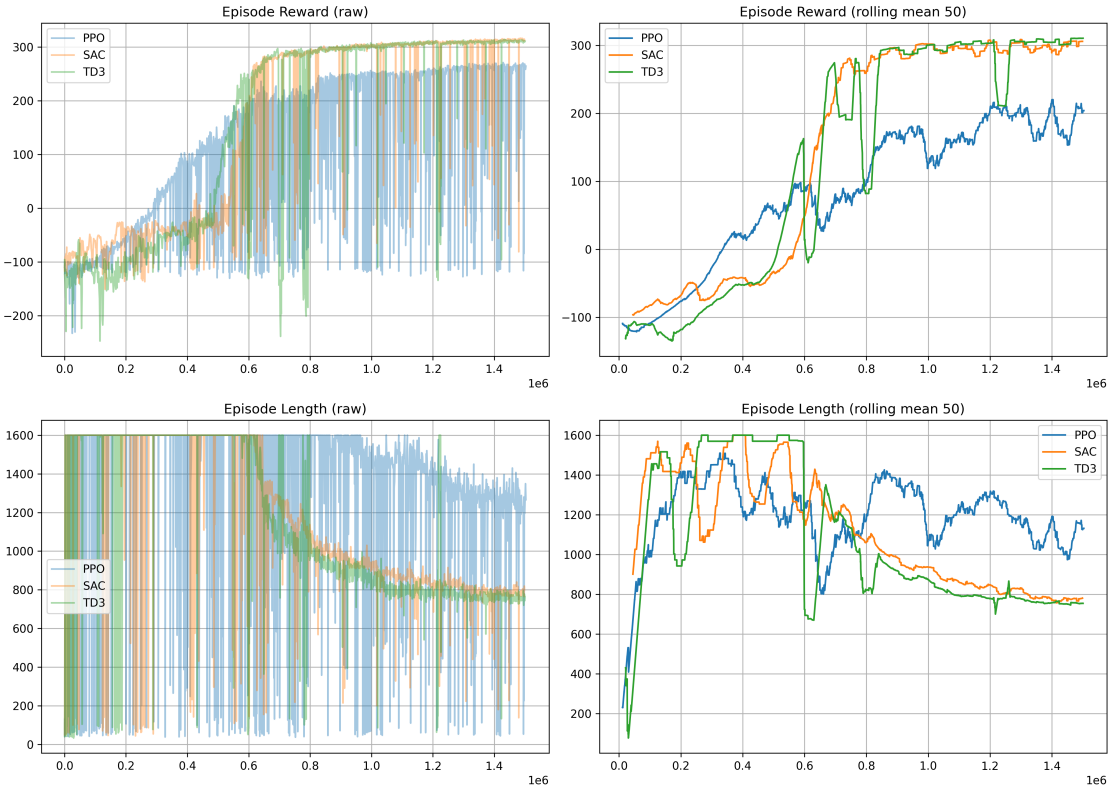


Figure 5.1: Comparison of PPO, SAC, and TD3 on BipedalWalker-v3: episode reward and episode length, shown in raw and smoothed form.

Figure 5.1 shows a single comprehensive plot composed of four subplots:

- Episode reward (raw),

- Episode reward (rolling mean with window size 50),
- Episode length (raw),
- Episode length (rolling mean with window size 50).

All curves are plotted as a function of the total number of environment timesteps, computed as the cumulative sum of episode lengths. PPO, SAC, and TD3 are displayed simultaneously to allow direct comparison.

## 5.2 Episode Reward Analysis

Episode reward provides a direct measure of the agent's ability to solve the locomotion task and is therefore a primary indicator of learning progress and policy performance.

### 5.2.1 Episode Reward (Raw)

The raw episode reward curves exhibit extremely high variance for all three algorithms. This behavior is characteristic of the BipedalWalker environment due to its stochastic dynamics, frequent falls, and high sensitivity to continuous action perturbations.

**PPO** PPO displays persistent variance throughout training, even after apparent convergence. Frequent negative reward spikes indicate repeated falls, suggesting that the learned policy remains unstable at the episode level. This reflects the sensitivity of on-policy updates and the limited reuse of past experience.

**SAC** SAC initially shows strong reward fluctuations, followed by frequent high-reward episodes. However, even after convergence, occasional failed episodes remain visible. This sustained variance is expected, as entropy-regularized exploration remains active during training.

**TD3** TD3 progresses more rapidly toward high rewards and exhibits a noticeable reduction in variance earlier than PPO. While occasional failures still occur, they become less frequent after convergence, indicating improved policy stability.

### 5.2.2 Episode Reward (Rolling Mean)

To enable meaningful comparison, a rolling mean with a window size of 50 episodes is applied. This smoothing reduces stochastic noise while preserving long-term learning trends, making it easier to identify learning speed, convergence behavior, and final performance levels.

**Learning Phase** TD3 demonstrates the fastest learning, reaching rewards close to 300 after approximately 0.7–0.8 million timesteps. This abrupt improvement highlights its strong sample efficiency.

SAC closely follows TD3, with slightly slower but very stable convergence. It also reaches the performance ceiling of the environment.

PPO learns significantly more slowly, with gradual improvements and no sharp transition. Its performance plateaus at a noticeably lower level, around 180–210.

**Convergence Phase** Both TD3 and SAC stabilize around rewards exceeding 300, close to the maximum achievable score for BipedalWalker. While minor temporary drops occur, both algorithms recover quickly.

PPO converges to a lower plateau and continues to exhibit fluctuations, reflecting the inherent instability of on-policy optimization in long-horizon continuous control tasks.

Overall, the final performance ranking is:

$$\text{TD3} \approx \text{SAC} > \text{PPO}.$$

## 5.3 Episode Length Analysis

Episode length reflects the agent's ability to maintain balance and avoid falls, and thus serves as a complementary indicator of locomotion stability.

### 5.3.1 Episode Length (Raw)

Episode length is strongly correlated with the stability of the walking behavior. Short episodes correspond to early falls, while long episodes indicate sustained locomotion.

All algorithms show high variance at the beginning of training. PPO continues to produce extremely short episodes even at later stages, indicating persistent instability. In contrast, SAC and TD3 exhibit a clearer transition toward consistently longer episodes.

### 5.3.2 Episode Length (Rolling Mean)

During the initial phase, all algorithms quickly achieve long episodes (approximately 1400–1600 steps), indicating that they learn to avoid immediate falls.

TD3 reaches the maximum episode length earlier than the other methods. A sharp drop around 0.6 million timesteps suggests a transient instability, followed by rapid recovery.

SAC maintains long episodes but displays more fluctuations after convergence, reflecting ongoing stochastic exploration.

PPO also achieves long episodes but shows weaker correlation between episode length and reward. The agent can survive for long durations while producing inefficient or energetically suboptimal locomotion.

This highlights an important distinction: long episodes do not necessarily imply high reward. PPO often prioritizes survival over efficient forward movement, whereas SAC and TD3 better optimize the full reward structure, including speed, stability, and energy usage.

## 5.4 Quantitative Summary

After 1.5 million timesteps, the average episode performance is summarized as follows:

- PPO: mean episode reward of 186 with a mean episode length of approximately 1070.
- SAC: mean episode reward of 306 with a mean episode length of approximately 777.

- TD3: mean episode reward of 306 with a mean episode length of approximately 750.

These results confirm that off-policy methods achieve significantly higher reward despite shorter episodes, indicating more efficient and goal-directed locomotion.

## 5.5 Qualitative Evaluation

To complement quantitative metrics, trained policies were visually inspected by generating animated rollouts. For each algorithm, a GIF was created by rendering environment frames while executing the learned policy deterministically.

This qualitative analysis confirms the numerical results: PPO exhibits cautious but inefficient walking patterns, while SAC and TD3 produce smoother, faster, and more stable gaits. Visual inspection further highlights the superior control quality achieved by off-policy algorithms.

## 5.6 Limitations and Perspectives

The results presented in this chapter highlight a clear performance gap between on-policy and off-policy reinforcement learning algorithms on the BipedalWalker environment. However, it is important to emphasize that the objective of this project was not to identify the optimal algorithm or hyperparameter configuration, but rather to study and compare representative reinforcement learning methods capable of successfully solving the task.

The choice of algorithms and hyperparameters was guided by commonly used configurations in the literature and practical experimentation, with the primary goal of achieving stable learning and reaching the environment's objective. No exhaustive hyperparameter optimization was performed. In particular, systematic approaches such as grid search, random search, or Bayesian optimization were not applied. It is therefore likely that improved or more stable performance could be obtained through more extensive tuning, alternative network architectures, or longer training horizons.

Additionally, all algorithms were trained with a fixed budget of 1.5 million environment timesteps. Increasing the total number of timesteps could further improve convergence, especially for PPO, which typically requires more on-policy data to reach competitive performance in complex continuous control tasks.

Beyond the algorithms evaluated in this work, several other reinforcement learning methods could be considered to potentially achieve better results. Examples include Trust Region Policy Optimization (TRPO) (Schulman et al. (2015)), which enforces stricter constraints on policy updates, or more recent actor-critic variants such as Asynchronous Advantage Actor-Critic (A3C) (Mnih et al. (2016)) and Distributed Distributional Deterministic Policy Gradients (D4PG) (Barth-Maron et al. (2018)). Model-based reinforcement learning approaches could also be explored to improve sample efficiency by leveraging learned environment dynamics.

Finally, combining algorithmic improvements with reward shaping, curriculum learning, or domain-specific regularization could further enhance locomotion quality and stability. These directions represent promising extensions of the present work, but fall outside the scope of this project.

# Chapter 6

# Conclusion

This project investigated the application of deep reinforcement learning algorithms to the continuous control problem of bipedal locomotion in the `BipedalWalker-v3` environment. Three representative actor-critic methods were implemented and evaluated: Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), and Twin Delayed Deep Deterministic Policy Gradient (TD3).

Through a systematic experimental setup, all algorithms were trained under comparable conditions and evaluated using episode reward and episode length as primary performance metrics. The results highlight clear differences in learning dynamics, stability, and final performance. Off-policy algorithms, namely SAC and TD3, consistently outperformed the on-policy PPO method in terms of both learning speed and achieved reward. In particular, TD3 demonstrated the highest sample efficiency, while SAC achieved comparable performance with increased robustness due to entropy regularization.

Beyond numerical performance, qualitative inspection of trained policies revealed distinct locomotion behaviors. PPO tended to favor conservative strategies that prioritize survival but often resulted in inefficient or unstable walking. In contrast, SAC and TD3 learned smoother and more effective gait patterns, better aligning with the full structure of the reward function, including forward progress, balance, and energy efficiency.

This study also emphasized the importance of careful experimental design in reinforcement learning. While the chosen hyperparameters and training budgets were sufficient to solve the task, no exhaustive optimization was performed. As discussed, improved results could potentially be obtained through extended training, systematic hyperparameter search, alternative architectures, or the evaluation of additional algorithms. Nevertheless, the primary objective of this project was to analyze representative reinforcement learning methods rather than to achieve state-of-the-art performance.

Overall, this work provides a practical comparison of on-policy and off-policy reinforcement learning algorithms on a challenging continuous control benchmark. It highlights the strengths of off-policy methods for locomotion tasks and reinforces the role of BipedalWalker as a meaningful environment for studying stability, exploration, and sample efficiency in deep reinforcement learning.

# References


Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N. and Lillicrap, T. (2018), Distributional policy gradients, *in* 'International Conference on Learning Representations'.

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', Vol. 80 of *Proceedings of Machine Learning Research*, PMLR, pp. 1587–1596.

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', Vol. 80 of *Proceedings of Machine Learning Research*, PMLR, pp. 1861–1870.

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, *in* '4th International Conference on Learning Representations, ICLR'.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K. (2016), Asynchronous methods for deep reinforcement learning, *in* 'Proceedings of The 33rd International Conference on Machine Learning', Vol. 48 of *Proceedings of Machine Learning Research*, PMLR, pp. 1928–1937.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M. and Dormann, N. (2021), 'Stable-baselines3: Reliable reinforcement learning implementations', *Journal of Machine Learning Research* **22**(268), 1–8.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. and Moritz, P. (2015), Trust region policy optimization, *in* 'Proceedings of the 32nd International Conference on Machine Learning', Vol. 37 of *Proceedings of Machine Learning Research*, PMLR, pp. 1889–1897.

Schulman, J., Moritz, P., Levine, S., Jordan, M. I. and Abbeel, P. (2016), High-dimensional continuous control using generalized advantage estimation, *in* '4th International Conference on Learning Representations, ICLR'.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017), 'Proximal policy optimization algorithms'.

Towers, M., Kwiatkowski, A., Balis, J. U., Cola, G. D., Deleu, T., Goulão, M., Andreas, K., Krimmel, M., KG, A., Perez-Vicente, R. D. L., Terry, J. K., Pierré, A., Schulhoff, S. V., Tai, J. J., Tan, H. and Younis, O. G. (2025), Gymnasium: A standard interface for reinforcement learning environments, *in* 'The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track'.

Watkins, C. J. C. H. and Dayan, P. (1992), 'Q-learning', *Machine Learning* **8**(3–4), 279–292.