Comparative Analysis of Deep Reinforcement Learning Algorithms for Bipedal Walker Locomotion

Author Name
Institution/University
Email: author@example.com

December 28, 2025

Abstract

Bipedal locomotion is a fundamental challenge in robotics and reinforcement learning, requiring agents to learn stable walking patterns on varied terrains. This paper presents a comprehensive comparative study of three state-of-the-art deep reinforcement learning algorithms—Twin Delayed Deep Deterministic Policy Gradient (TD3), Proximal Policy Optimization (PPO), and Soft Actor-Critic (SAC)—for training bipedal walking agents. We evaluate these algorithms on two environment variants: standard terrain (without obstacles) and hardcore mode (with diverse obstacles including stairs, pits, and holes). Our study aims to determine which algorithm achieves the best performance in terms of convergence speed, stability, and robustness across different terrain difficulties. Detailed experimental analysis includes hyperparameter configurations, learning curves, and performance metrics. The results provide practitioners with empirical guidance for algorithm selection in continuous control tasks with varying environmental complexity.

1 Introduction

1.1 Problem Definition

The bipedal walker task, formalized as a continuous control problem, requires an agent to learn locomotion policies that enable stable walking across varied terrain. This problem is challenging due to several factors:

- **High-dimensional action space**: Continuous control of multiple joint angles and forces.
- **Complex dynamics**: Non-linear interactions between agent movements and environmental physics.
- **Sample efficiency**: Limited budget of environment interactions during training.
- **Robustness**: Generalization to unseen or more difficult terrain configurations.

1.1.1 Action Space

The action space consists of 4 continuous values normalized in the range $[-1, 1]$:

- Action 0: Right hip motor speed control
- Action 1: Right knee motor speed control
- Action 2: Left hip motor speed control
- Action 3: Left knee motor speed control

Each action controls the motor torque magnitude and direction. The maximum motor torque is set to 80 units, with hip motors capable of speeds up to 4 rad/s and knee motors up to 6 rad/s.

1.1.2 Observation Space

The observation vector consists of 24 continuous features representing the agent's state:

Hull state (3 features):

- Hull angle relative to vertical (radians, normalized to $[-\pi, \pi]$)
- Hull angular velocity (normalized, range $[-5, 5]$)

Velocity state (2 features):

- Horizontal velocity normalized by viewport width
- Vertical velocity normalized by viewport height

Joint states (8 features):

- 4 joint angles (hip and knee angles for both legs, normalized)
- 4 joint angular velocities (normalized by motor speed constants)

Contact flags (2 features):

- Right leg ground contact (binary: 0 or 1)
- Left leg ground contact (binary: 0 or 1)

Lidar measurements (10 features):

- 10 distance measurements from lidar-like rangefinders distributed around the agent
- Each measurement returns the fraction of the lidar range that hits terrain (0 to 1)

Total observation dimension: 24 continuous values.

1.1.3 Reward Function

The reward structure incentivizes forward progress while penalizing energy consumption and failures. At a high level, the per-step reward can be decomposed as:

$$r_t = r_t^{\text{progress}} + r_t^{\text{posture}} + r_t^{\text{energy}} + r_t^{\text{fail}}.$$

where:

- r_t^{progress} encourages forward position progress of the agent.
- r_t^{posture} penalizes large deviations of the hull angle to keep the torso roughly upright.
- r_t^{energy} penalizes the magnitude of motor torques (energy usage).
- r_t^{fail} applies a large negative penalty when the hull touches the ground (agent has fallen).

The exact implemented reward formula, including all coefficients and normalization constants, is given in Section 3.1.3.

Success criteria:

- Standard mode: Achieve 300+ total reward within 1600 timesteps
- Hardcore mode: Achieve 300+ total reward within 2000 timesteps

1.2 Main Contributions

This study contributes the following:

1. **Comprehensive empirical comparison** of three popular reinforcement learning algorithms (TD3, PPO, SAC) on the bipedal walker task under two difficulty levels.
2. **Systematic evaluation framework** including standardized hyperparameter sets, training protocols, and evaluation metrics.
3. **Performance analysis** across multiple dimensions: convergence speed, solution stability, reward distribution.

1.3 Approach Overview

We train and evaluate three reinforcement learning algorithms on the BipedalWalker-v3 environment under two configurations:

- **Standard mode**: Flat terrain with moderate difficulty.
- **Hardcore mode**: Complex terrain with obstacles, stairs, pits, and holes.

For each algorithm and environment configuration, we conduct multiple training runs with fixed random seeds to ensure reproducibility. We measure performance using episode rewards, training convergence curves, evaluation metrics, and sample efficiency. This enables a fair comparison of algorithm effectiveness in both benign and challenging conditions.

2 Related Work

2.1 Reinforcement Learning for Continuous Control

Deep reinforcement learning has emerged as a powerful approach for solving continuous control problems. Early work by Mnih et al. [**?**] introduced Deep Q-Networks (DQN) for discrete action spaces, which later inspired continuous control methods.

2.2 Policy Gradient Methods

Proximal Policy Optimization (PPO) is an on-policy algorithm that improves upon prior policy gradient methods by using a clipped surrogate objective to prevent overly large policy updates [**?**]. PPO is known for its stability and ease of implementation, making it a popular baseline in continuous control research.

2.3 Off-Policy Actor-Critic Methods

Deep Deterministic Policy Gradient (DDPG) [**?**] introduced off-policy learning for continuous control by combining actor-critic methods with experience replay. However, DDPG suffers from Q-function overestimation issues that can lead to training instability.

Twin Delayed Deep Deterministic Policy Gradient (TD3) [**?**] addresses DDPG's overestimation by maintaining twin critics and using target smoothing regularization, resulting in improved stability and sample efficiency.

Soft Actor-Critic (SAC) [**?**] combines off-policy learning with entropy regularization to encourage exploration and robustness. SAC maximizes both expected return and policy entropy, leading to better exploration and improved performance on diverse tasks.

2.4 Comparison with Prior Work

Our study differs from prior work in several ways:

- **Multiple difficulty levels**: We evaluate algorithms on both standard and hardcore environments, testing generalization to harder terrain.
- **Systematic hyperparameter study**: We provide detailed hyperparameter configurations for each algorithm optimized for this task.
- **Reproducibility focus**: All experiments are conducted with fixed random seeds and multiple training runs for statistical reliability.

3 Environment and Task Setup

3.1 BipedalWalker-v3 Environment

The BipedalWalker-v3 environment [**?**] is a classic benchmark for continuous control in reinforcement learning. It simulates a 4-jointed bipedal robot that must learn to walk forward while minimizing energy consumption.

3.1.1 Observation Space

The agent receives observations comprising:

- Hull angle, angular velocity, and velocity (3 dimensions)
- Position and velocity of 4 legs (8 dimensions)
- Contact flags for leg-ground interactions (4 dimensions)
- Lidar-like distance measurements to terrain ahead (10 dimensions)

Total observation dimension: 24 features.

3.1.2 Action Space

The action space consists of 4 continuous values in the range $[-1, 1]$, corresponding to motor torques for:

- Hip actuators (2 motors)
- Knee actuators (2 motors)

3.1.3 Reward Function

The reward is defined as:

$$r_t = v_x - 0.005 \cdot \left(\sum_i a_i^2 \right) - \mathbb{1}_{\text{fell}}$$

where:

- v_x is forward velocity (reward for progress)
- $\sum_i a_i^2$ is the sum of squared actions (penalizes energy consumption)
- $\mathbb{1}_{\text{fell}}$ indicates episode failure (agent falls = -1)

3.2 Environment Variants

3.2.1 Standard Mode

Clean, relatively flat terrain with gentle variations. This mode tests basic locomotion capability.

3.2.2 Hardcore Mode

Challenging terrain featuring:

- Stairs and descents
- Pits and gaps
- Various obstacles
- Uneven surfaces

This mode tests robustness and generalization to complex environments.

4 Methodology

4.1 Reinforcement Learning Algorithms

4.1.1 TD3 (Twin Delayed DDPG)

TD3 is an off-policy actor-critic algorithm that improves upon DDPG through three mechanisms:

1. **Twin Critics**: Two Q-functions Q_{θ_1} and Q_{θ_2} provide estimates. The target uses the minimum: $\min(Q_{\theta_1}, Q_{\theta_2})$.

2. **Delayed Policy Updates**: The actor is updated less frequently than critics (every k critic updates), reducing overestimation bias propagation.

3. **Target Smoothing Regularization (TSMR)**: Target actions are perturbed with clipped noise:

$$a' = \text{clip}(\mu_{\phi'}(s') + \epsilon, -c, c), \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

The Bellman target is:

$$y = r + \gamma \min_{i \in \{1,2\}} Q_{\theta_i'}(s', a')$$

4.1.2 PPO (Proximal Policy Optimization)

PPO is an on-policy algorithm that optimizes the policy using a clipped surrogate objective:

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

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio and $\hat{A}_t$ is the advantage estimate computed using Generalized Advantage Estimation (GAE).

4.1.3 SAC (Soft Actor-Critic)

SAC is an off-policy algorithm that maximizes a trade-off between expected return and policy entropy:

$$J(\pi) = \mathbb{E}_{s \sim \mathcal{D}} \left[\mathbb{E}_{a \sim \pi}[Q(s,a) - \alpha \log \pi(a|s)] \right]$$

The algorithm maintains:

- Stochastic policy $\pi_\phi(a|s)$ (Gaussian)
- Two Q-networks for overestimation mitigation
- Optional automatic entropy temperature tuning

4.2 Training Setup

4.2.1 Network Architecture

All algorithms use identical network architectures for fair comparison:

- **Policy Network**: MLP with hidden layers [256, 256], ReLU activations, tanh output for actions.
- **Value Networks**: MLP with hidden layers [256, 256], ReLU activations, linear output.

4.2.2 Hyperparameters

The hyperparameter configurations for each algorithm are presented in the results section.

4.3 Training Protocol

1. Initialize environment and agent with fixed seed.
2. Collect experience through environment interaction.
3. Update agent parameters periodically.
4. Evaluate performance every 10k steps on 10 evaluation episodes.
5. Repeat until convergence or maximum timesteps reached.

4.4 Evaluation Metrics

- **Episode Return**: Sum of rewards over an episode.
- **Mean Reward (Last 100 Episodes)**: Rolling average to assess convergence.
- **Convergence Speed**: Training steps to reach target performance threshold.
- **Stability**: Standard deviation of evaluation rewards.
- **Sample Efficiency**: Reward achieved per environment interaction.

5 Experiments and Results

5.1 Experimental Setup

5.1.1 Hardware and Software

- **Python Version**: 3.x
- **Framework**: PyTorch
- **Environment**: BipedalWalker-v3 (OpenAI Gym)
- **Device**: [CPU/GPU specification]

5.1.2 Training Runs

[Number of training runs per algorithm/environment combination, random seed specifications, total training timesteps]

5.2 Hyperparameter Configurations

5.2.1 TD3 Configuration

Parameter	Value
Learning Rate	[VALUE]
Gamma (Discount Factor)	[VALUE]
Tau (Soft Update)	[VALUE]
Target Noise	[VALUE]
Noise Clip	[VALUE]
Policy Update Frequency	[VALUE]
Batch Size	[VALUE]
Replay Buffer Capacity	[VALUE]

Table 1: TD3 hyperparameter configuration

5.2.2 PPO Configuration

Parameter	Value
Learning Rate	[VALUE]
Gamma (Discount Factor)	[VALUE]
GAE Lambda	[VALUE]
Clip Epsilon	[VALUE]
Value Loss Coefficient	[VALUE]
Entropy Coefficient	[VALUE]
PPO Epochs	[VALUE]
Mini-batch Size	[VALUE]
Rollout Steps	[VALUE]

Table 2: PPO hyperparameter configuration

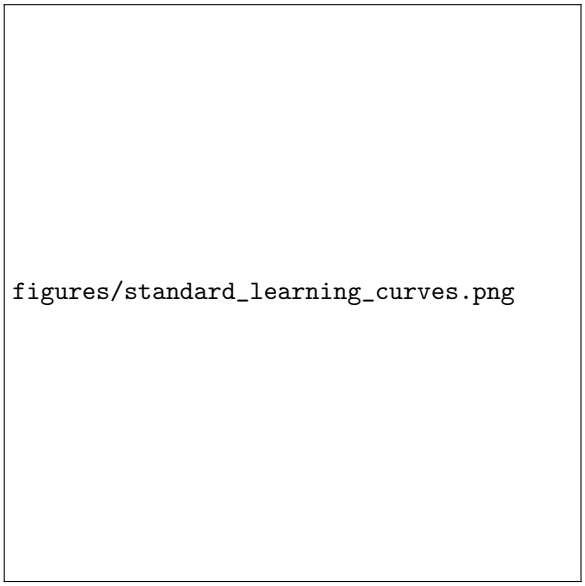
5.2.3 SAC Configuration

Parameter	Value
Learning Rate	[VALUE]
Gamma (Discount Factor)	[VALUE]
Tau (Soft Update)	[VALUE]
Alpha (Entropy Coefficient)	[VALUE]
Automatic Entropy Tuning	[VALUE]
Batch Size	[VALUE]
Replay Buffer Capacity	[VALUE]
Learning Starts	[VALUE]

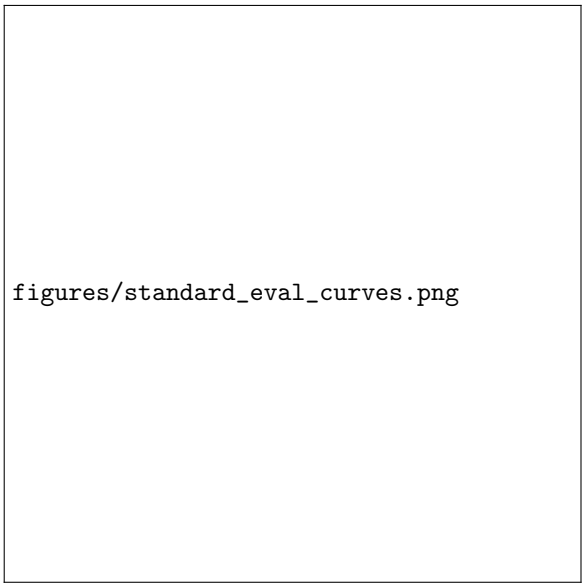
Table 3: SAC hyperparameter configuration

5.3 Performance Results

5.3.1 Standard Mode Results



(a) Episode rewards vs training steps



(b) Evaluation rewards vs training steps

Figure 1: Standard Mode: Learning Progress

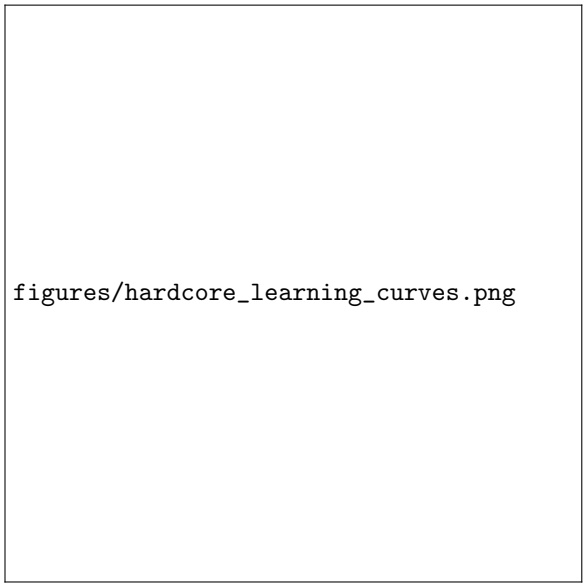
Learning Curves [Space for figure or description of results]

Metric	TD3	PPO	SAC
Final Mean Reward	[VALUE]	[VALUE]	[VALUE]
Convergence Steps	[VALUE]	[VALUE]	[VALUE]
Reward Std Dev	[VALUE]	[VALUE]	[VALUE]
Sample Efficiency	[VALUE]	[VALUE]	[VALUE]

Table 4: Standard Mode: Comparative Performance Metrics

Quantitative Results [Space for analysis and interpretation]

5.3.2 Hardcore Mode Results



(a) Episode rewards vs training steps

(b) Evaluation rewards vs training steps

Figure 2: Hardcore Mode: Learning Progress

Learning Curves [Space for figure or description of results]

Metric	TD3	PPO	SAC
Final Mean Reward	[VALUE]	[VALUE]	[VALUE]
Convergence Steps	[VALUE]	[VALUE]	[VALUE]
Reward Std Dev	[VALUE]	[VALUE]	[VALUE]
Sample Efficiency	[VALUE]	[VALUE]	[VALUE]

Table 5: Hardcore Mode: Comparative Performance Metrics

Quantitative Results [Space for analysis and interpretation]

5.4 Comparative Analysis

Algorithm	Std Mode	Hard Mode	Stability	Sp
TD3	[RANKING]	[RANKING]	[RANKING]	[RAN
PPO	[RANKING]	[RANKING]	[RANKING]	[RAN
SAC	[RANKING]	[RANKING]	[RANKING]	[RAN

Table 6: Overall Algorithm Rankings

Algorithm Performance Summary [Space for detailed comparative discussion]

6 Discussion

6.1 Key Findings

[Space for discussion of main results]

- Finding 1: [Description and implications]
- Finding 2: [Description and implications]
- Finding 3: [Description and implications]

6.2 Algorithm-Specific Observations

6.2.1 TD3

[Analysis of TD3's performance, strengths, and weaknesses]

6.2.2 PPO

[Analysis of PPO's performance, strengths, and weaknesses]

6.2.3 SAC

[Analysis of SAC's performance, strengths, and weaknesses]

6.3 Difficulty Transfer

[Discussion of how algorithms perform when transitioning from standard to hardcore mode]

6.4 Sample Efficiency and Computational Cost

[Comparison of sample efficiency and computational requirements]

7 Conclusion

This study presents a comprehensive empirical comparison of three state-of-the-art deep reinforcement learning algorithms—TD3, PPO, and SAC—on the bipedal walker locomotion task across two difficulty levels. Our experiments provide systematic evidence of each algorithm's effectiveness in continuous control with varying environmental complexity.

7.1 Summary

[Summary of main findings]

7.2 Practical Implications

[Guidance for practitioners on algorithm selection based on task requirements]

7.3 Limitations

- Limited to single benchmark environment

Additional limitations

7.4 Future Work

1. Extend evaluation to additional continuous control benchmarks
2. Investigate hybrid approaches combining strengths of multiple algorithms
3. Study transfer learning and fine-tuning across related tasks
4. Analyze computational efficiency and scalability
5. Conduct ablation studies on key hyperparameters

8 References

References

[1] Mnih, V., Kavukcuoglu, K., & Silver, D. (2013). "Playing Atari with Deep Reinforcement Learning." *arXiv preprint arXiv:1312.5602*.

[2] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). "Proximal Policy Optimization Algorithms." *arXiv preprint arXiv:1707.06347*.

[3] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). "Continuous control with deep reinforcement learning." *arXiv preprint arXiv:1509.02971*.

[4] Fujimoto, S., Hoof, H., & Meger, D. (2018). "Addressing Function Approximation Error in Actor-Critic Methods." *International Conference on Machine Learning (ICML)*.

[5] Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor." *International Conference on Machine Learning (ICML)*.

[6] Brockman, G., Cheung, V., Petrov, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). "OpenAI Gym." *arXiv preprint arXiv:1606.01540*.

A Additional Experimental Details

[Space for additional experimental configuration details if needed]

B Extended Results Tables

[Space for extended results and statistical analysis]