Mastering Continuous Control: Comparing A2C, SAC, and TD3 in OpenAI Gym

Sophie Pavia
Department of Computer Science
Vanderbilt University
Nashville, TN 37203
`sophie.r.pavia@vanderbilt.edu`

Abstract

This project evaluates the performance of three reinforcement learning (RL) methods: Advantage Actor-Critic (A2C), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Soft Actor-Critic (SAC) in continuous action OpenAI Gym environments, Pendulum and BipedalWalker. The focus is on comparing the on-policy (A2C) and off-policy methods (TD3 and SAC) in terms of sample efficiency, learning stability, and robustness in challenging continuous control tasks. We investigate how SAC's entropy-based exploration compares to TD3's targeted action-value optimization and A2C's synchronous policy updates. Additionally, we analyze how these methods perform in environments of varying complexity, from the simpler Pendulum task to the more dynamic and high-dimensional BipedalWalker. We aim to provide insights into the trade-offs between one on-policy and two off-policy approaches in continuous-action domains, highlighting their strengths and weaknesses under different environments.

1 Introduction

Reinforcement learning (RL) can be famously described as learning how to map situations to actions to maximize a numerical reward signal (Sutton and Barto [2018].) An agent, or learner, must discover which actions by trying them. This can be difficult to do in large action spaces, particularly in continuous action spaces with infinite actions. This project investigates continuous-control tasks using different reinforcement learning algorithms in OpenAI Gym environments. The specific focus of this work is to evaluate and compare three reinforcement learning methods: Advantage Actor-Critic (A2C), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Soft Actor-Critic (SAC) in terms of sample efficiency, learning stability, and robustness. We survey and explore the algorithms' performance on continuous-control tasks to provide insights into the trade-offs between on-policy and off-policy reinforcement learning methods, shedding light on their advantages and limitations across tasks with varying complexities.

Importance and Interest: One of the primary goals of the field of artificial intelligence is to develop systems capable of learning and solving complex tasks with human-like adaptability, particularly in high-dimensional, uncertain environments. Reinforcement learning can help achieve this goal by developing agents that can learn optimal decision-making policies through interaction with their environment. However, traditional RL methods often struggle in large action spaces because they are challenging to model and solve effectively. Several methods have been developed to address continuous actions, such as policy parametrization and other suitable techniques; however, this is a topic of ongoing research with as yet no clear resolution.

To bridge this gap, environments like OpenAI Gym (OpenAI [2016]) provide a platform for experimenting with toy problems in continuous action spaces. These environments serve as proxies for

Preprint. Under review.

real-world tasks, allowing researchers to explore the strengths and limitations of RL algorithms in a controlled setting. It is important to consider that experimenting with these RL methods in OpenAI Gym environments does not address the sim-to-real gap. The sim-to-real gap is a significant challenge in RL, where methods that perform well in simulated environments often fail to generalize effectively to real-world applications. This is an ongoing research area focused on achieving human-like learning capabilities outside of simulation environments. It is an important issue in RL to highlight, even though this work does not focus on it for the algorithms and environments analyzed in the report.

Approach Overview: We implement three RL methods using a popular RL library, StableBaselines3 (Raffin et al. [2021]), and evaluate them on two OpenAI Gym Environments to compare each algorithm's performance based on different performance metrics. We consider different characteristics of the algorithms in our evaluation. Specific algorithm characteristics can be found in Section 3.2

Report Organization: The remainder of this paper is organized as follows. Section 2 provides background information, including a review of state-of-the-art reinforcement learning approaches and related work. Section 3 outlines the methodology, detailing the approach, algorithms, and implementation used in this paper. Section 4 presents the experimental setup, results, and a critical analysis of the findings. Finally, Section 5 discusses the implications of the results, compares them with related work, and highlights directions for future research.

2 Background

Many real-world problems have continuous state or action spaces, both of which make learning good policies difficult. RL methods like PPO, A2C, DDPG, SAC, and TD3 have shown to be powerful tools for solving continuous control tasks (Fujimoto et al. [2018], Haarnoja et al. [2018], Mnih et al. [2016], Lillicrap et al. [2019], Schulman et al. [2017]). In this paper we examine A2C, TD3, and SAC which are actor-critic frameworks, more details about these algorithms can be found in Section 3.2. A2C is the synchronous version of A3C, which was developed by Mnih et al. [2016]. An approach that enables the use of an approximate value function to estimate the policy gradient was presented by Sutton et al. [1999] and then OpenAI presented a baseline implementation of A2C (Wu et al. [2017]). A2C is a classic policy gradient method that improves stability and sample efficiency compared to earlier approaches like REINFORCE (Williams [1987, 1992]). However, A2C remains limited in complex environments due to its on-policy nature.

To address these challenges, off-policy algorithms like SAC and TD3 were developed. SAC incorporates entropy regularization to balance exploration and exploitation, enabling robust learning in environments with high-dimensional action spaces. TD3 improves upon DDPG by mitigating overestimation bias with different techniques. These algorithms have become extensively tested and used as baselines to solve different environments such as those provided by OpenAI Gym. We discuss the algorithms and environments in detail in Section 3.

2.1 Related Work

Reinforcement Learning for continuous control tasks has presented several challenges as the field evolves, including sample complexity, high-dimensional state-action spaces, stability of learning, and notably reproducibility (Dulac-Arnold et al. [2019]). Frameworks such as OpenAI Gym (Brockman et al. [2016]) and libraries such as Stable Baselines3 (Raffin et al. [2021]) serve as critical tools for benchmarking and addressing these challenges.

Sample Complexity: Sample efficiency is a significant concern for RL methods, as training can often require vast amounts of data. On-policy methods like A2C are less sample-efficient since they discard past experiences after each update. This limitation can hinder exploration and slow convergence, especially in complex environments (Schulman et al. [2017]). As previously mentioned, off-policy methods can alleviate this. Algorithms such as SAC and TD3 leverage experience replay buffers that reuse past transitions, which can improve sample efficiency (Haarnoja et al. [2018], Fujimoto et al. [2018]).

Stability and Reproducibility: Stable and reproducible implementations are critical for benchmarking RL algorithms and ensuring fair comparisons. Stable Baselines3 and RLlib (Liang et al. [2018]) are two widely used libraries that address these challenges through high-quality implementations and robust evaluation tools. Reproducibility in reinforcement learning (RL) has received significant attention, particularly in the context of hyperparameter optimization, due to RL's sensitivity to hyperparameter settings. The development of standardized metrics and rigorous experimental reporting practices has emerged as a critical focus in the field (Bischl et al. [2021], Henderson et al. [2018], Parker-Holder et al. [2022]). Although this is a significant area of study, a detailed discussion is beyond the scope of this work.

Stable Baselines3 (SB3): SB3 plays a vital role in addressing RL challenges by providing high-quality, reproducible implementations of state-of-the-art RL algorithms including A2C, SAC, and TD3. This library emphasizes code simplicity, reproducibility, and testing. It is a structured API, inspired by packages like scikit-learn, which allows for easy integration with OpenAI Gym. It allows for the support of vectorized environments, logging, and custom callbacks; all of which support reproducible experimentation and performance monitoring.

RLlib: RLlib is another reinforcement learning library built on the Ray framework (Moritz et al. [2018]) that provides a scalable, high-performance framework for building and training RL models. RLlib exceeds in large-scale RL computations and provides better scalability than SB3. However, it is a less intuitive API than SB3. Another notable difference is the algorithm support provided by RLlib versus SB3. Both SB3 and RLlib support fundamental algorithms such as DQN, PPO, and SAC, however, RLlib provides other more advanced algorithms such as IMPALA (Espeholt et al. [2018]) and MARWIL (Wang et al. [2018]). RLlib also supports integration with OpenAI Gym as well as other environments such as PettingZoo (Terry et al. [2021]). Its scalability and flexibility make RLlib a strong choice for applications requiring distributed RL or large-scale simulations.

Together, Stable-Baselines3 and RLlib provide solutions for stability and reproducibility challenges within RL. SB3 prioritizes ease of use, reliability, and high-quality implementations, making it well-suited for rapid experimentation and smaller-scale continuous control tasks. RLlib is optimized for scalability and distributed RL, enabling efficient training in more computationally demanding settings.

3 Approach

3.1 Overview of the Approach

This paper's approach focuses on evaluating the performance of Advantage Actor-Critic (A2C), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Soft Actor-Critic (SAC) on two OpenAI Gym continuous action-space environments: Pendulum and BipedalWalker. These environments serve as controlled benchmarks for understanding algorithm behavior in tasks of increasing complexity. The Pendulum environment represents a low-dimensional control problem where the goal is to swing and balance a pendulum upright, serving as a simpler benchmark. In contrast, the BipedalWalker environment introduces higher complexity, requiring a simulated bipedal robot to learn stable walking, making it a challenging task with dynamic, high-dimensional states and action spaces. When evaluating the algorithms in these environments we will consider the different characters of each algorithm that might lead to differences in performance. For example, we might consider how TD3 addresses function approximation error in actor-critic methods by delaying target updates and employing clipped double Q-learning, while SAC builds on TD3 by incorporating the maximum entropy framework and leveraging the double Q-learning trick for enhanced exploration and stability. In the following section, we will go into detail about the different characteristics of each algorithm considered.

3.2 Algorithm and Implementation Details

A2C, SAC, and TD3 are implemented in Python using the Stable Baselines3 (Raffin et al. [2021]) library which is built on PyTorch. TD3 has an integrated noise model based on the hyperparameter 'action noise.' This model is used to handle exploration in continuous action spaces. All algorithms were implemented with hyperparameters from RL Baselines3 Zoo. See Section 3.4 for more details on hyperparameter optimization. Table 1 categorizes each of these algorithms in terms of key features.

Advantage Actor-Critic (A2C): This is a policy gradient algorithm that is an extension of basic actor-critic architecture. A2C is a synchronous, deterministic variant of The Asynchronous Advantage Actor Critic (A3C). To eliminate the need for a replay buffer, it employs several workers. A3C was

Table 1: Comparison of A2C, TD3, and SAC for Continuous Control Tasks

Algorithm	**Type**	**On/Off-Policy**	**Deterministic/Stochastic**
A2C	Actor-Critic, Policy Gradient	On-Policy	Stochastic
TD3	Actor-Critic, Deterministic Policy Gradient	Off-Policy	Deterministic
SAC	Actor-Critic, Policy Gradient	Off-Policy	Stochastic

proposed by Mnih et al.. A2C's main difference from the basic actor-critic algorithm is that the advantage function replaces the discounted cumulative reward from policy gradients. Instead of having the critic learn the Q-values, the critic learns the advantage values. An advantage function captures how better an action is compared to the others at a given state. OpenAI's A2C baseline has been shown to provide equal performance to A3C, as well as be more cost-effective than A3C when using single-GPU machines or a CPU-only implementation with larger policies (Wu et al. [2017]). A2C is well suited for continuous control tasks with dynamic environments like Pendulum and BipedalWalker because it optimizes stochastic policies. A stochastic policy can aid in the exploration of these types of environments by allowing the agent to try out different actions during training.

Twin Delayed DDPG (TD3): This off-policy algorithm addresses function approximation error in actor-critic methods and is a direct descendant of Deterministic Policy Gradient (DDPG). It approves upon DDPG by utilizing clipped double Q-learning, delayed policy update, and target policy smoothing with a pair of critics and a single actor Fujimoto et al. [2018]. It has been shown that function approximation errors can lead to overestimated value estimates and suboptimal policies. This problem appears in the actor-critic settings, therefore TD3 was created to minimize this effect on both the actor and the critic for continuous control.

Soft Actor Critic (SAC): This is an off-policy actor-critic deep RL algorithm that is based on the maximum entropy reinforcement learning framework (Haarnoja et al. [2018]). It was created on the idea that the use of stochastic policies and entropy maximization would be more stable and have better exploration than deep deterministic policy gradient (DDPG). The central feature of this algorithm is entropy regularization, where the policy is trained to maximize a trade-off between expected return and entropy. SAC also uses the clipped double Q-learning trick utilized by TD3, however it differs from TD3 with its use of entropy regularization and a stochastic policy. By combining off-policy updates with stable stochastic actor-critic formulation, this method has high performance on continuous control tasks. This approach has also proven to be very stable, achieving similar performance with different seeds.

3.3 Environments

The training and evaluation environments are from OpenAI Gymnasium (OpenAI [2016]) with vectorized environments to stack multiple independent environments into a single environment. This allows us to train n environments per step.

Pendulum: This environment is a classic control environment. It has a continuous action space and a continuous observation space. The action space is a ndarray that represents the torque applied to the free end of the pendulum. The observation space is also a ndarry that represents the x-y coordinates of the pendulum's free end and its angular velocity. The goal of this environment is to swing a pendulum to an upright position and keep it balanced. The control input to this task is torque applied to the free of the fixed pole. The reward function penalizes the pendulum for straying from the upright position, having a high angular velocity, and large control inputs. Therefore, the maximum value of the reward is 0, which can only occur when the pendulum is perfectly upright with no angular velocity or control effort. It has a minimum possible reward of approximately -16.27. The initial state of this environment is a random angle within the range of the observation state, as well as a random angular velocity within the range of the observation space. Episodes truncate after 200 time steps.

Bipedal Walker: This environment is a Box2D environment. Box2D is a type of environment for toy games based around physics control using box2d based physics (Catto [2011]). A 4-joint bipedal robot is simulated to walk across uneven terrain. There are two versions of this environment, including normal and hardcore. Hardcore differs from the normal uneven terrain by including ladders, stumps, and pitfalls. Like the Pendulum environment, it has a continuous action space and continuous observation space. The action space consists of motor speed values for 4 joints. The observation space is a 24-dimensional continuous state vector that consists of hull angle speed, angular velocity, horizontal speed, vertical speed, the position of joints and joints angular speed, legs contact with ground, and 10 LiDAR rangefinder measurements. There are no coordinates in the state vector. To solve the normal version, the robot must get a reward of 300 in 1600 time steps. The hardcore version is considered solved if receiving 300 points in 2000 time steps. A robot gets rewards as it moves forward, totaling at least 300 points for traveling to the far end. The robot will be negatively impacted by 100 points if it falls and also will experience a small decrease in points for applying motor torque. Termination of the episode occurs when the hull, or top of the robot, comes in contact with the ground or if the robot reaches the end of the terrain length.

3.4 Hyperparameters

Hyperparameter values for each algorithm were selected based on the environment used, referencing RL Baselines3 Zoo (Raffin [2020]). Experiments were not conducted for the hyperparameter selection because RL Baselines3 Zoo had tuned hyperparameters available for all algorithm environment pairs considered. This library used Optuna (Akiba et al. [2019]) for hyperparameter optimization, a popular hyperparameter optimization framework used to automate hyperparameter search. The hyperparameters YAML files from RL Baseline3 Zoo are uploaded and used during training.

4 Experiments and Results

4.1 Experiment Design

For each algorithm-environment pair, training is conducted using hyperparameters from RL Baselines3 Zoo. The models were implemented with callback mechanisms to stop training upon reward thresholds for BipedalWalker and a custom callback to log sample efficiency during training. All experiments used deterministic seeds for reproducibility and to compare robustness in terms of performance across different seeds.

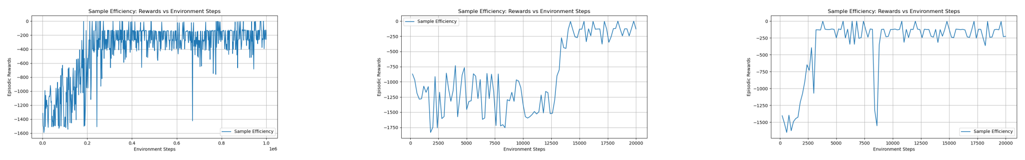
The trained models are then evaluated over 100 independent test episodes for each environment. We evaluate several metrics during training depending on the algorithm used and also measure the final performance as the total rewards achieved during testing. The learning stability, which reflects the consistency of reward progression over training steps, is also considered. Training logs, sample efficiency data, and models are saved for post-analysis, with visualizations generated by Tensorboard and Matplotlib to analyze trends.

Pendulum Environment: For training and evaluation, we use Pendulum-v1 with the default acceleration of gravity ($10ms^{-2}$).

Bipedal Walker Environment: BipedalWalker-v3 is used with the normal version. An event callback is used for this environment to enforce early stopping on a mean reward threshold of 300 points during training.

Table 2: Sample efficiency results for Pendulum.

Algorithm	Avg Reward	Max Reward	Min Reward	Timesteps to -250
A2C	-415.95	-0.63	-1588.90	184,000
TD3	-896.87	-0.71	-1833.13	14,000
SAC	-336.45	-0.70	-1652.43	3,200



(a) A2C (b) TD3 (c) SAC

Figure 1: Sample Efficiency in Pendulum Environment

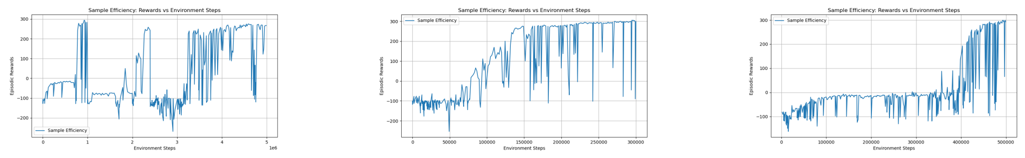
4.2 Results

We begin by presenting the results of the three RL algorithms on each continuous control task. The results are evaluated on several metrics, including sample efficiency, learning stability, evaluation reward, and robustness. Each algorithm's performance is analyzed to highlight its strengths and weaknesses in different environments to determine its effectiveness in continuous control tasks of varying degrees.

4.2.1 Training

Table 3: Sample efficiency results for BipedalWalker.

Algorithm	Avg Reward	Max Reward	Min Reward	Timesteps to 250
A2C	-22.71	294.26	-268.65	800,208
TD3	33.67	306.15	-253.55	136,400
SAC	-14.83	299.35	-161.89	423,743



(a) A2C (b) TD3 (c) SAC

Figure 2: Sample Efficiency in BipedalWalker Environment

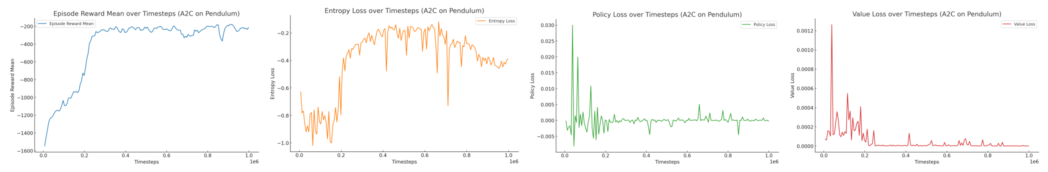
Sample Efficiency: Sample efficiency is assessed by evaluating the amount of experience required for each algorithm to achieve a certain level of performance.

Pendulum: Figure 1 shows the sample efficiency plotted as the reward progression over training timesteps for all three algorithms. SAC has rapid convergence to the performance threshold, while A2C is much slower and noisier learning. We also highlight the average, maximum, and minimum reward received during training in Table 2. We choose a performance threshold of -250 which indicates that the pendulum is being controlled reasonably well, albeit not perfectly upright. SAC has an advantage in not only learning time but also average learning reward. This advantage could be due to leveraging entropy-based exploration and off-policy updates. The other off-policy algorithm, TD3 also showed better compared to A2C which could be due to its ability to address function approximation errors through clipped double Q learning and delayed updates. These findings suggest that for simple continuous control tasks like Pendulum, off-policy algorithms such as SAC and TD3 are more sample-efficient compared to on-policy methods like A2C.

BipedalWalker: For the more complex BipedalWalker task, off-policy methods demonstrate superior sample efficiency. We chose a performance threshold of 250 because the maximum possible reward is around 300 for a robot crossing the terrain. A value between 200 and 300 indicates that the agent is performing well and can navigate the terrain effectively. As shown in Table 3 and Figure 2, TD3 reaches the performance threshold the fastest, benefiting from clipped double Q-learning and delayed policy updates. SAC follows at 423,743 timesteps, with stable rewards due to its entropy-regularized exploration. In contrast, A2C lags significantly, requiring 800,208 timesteps with noisier learning and lower average rewards. These findings confirm that off-policy algorithms like TD3 and SAC are far more efficient and robust than on-policy methods like A2C for complex continuous control tasks.

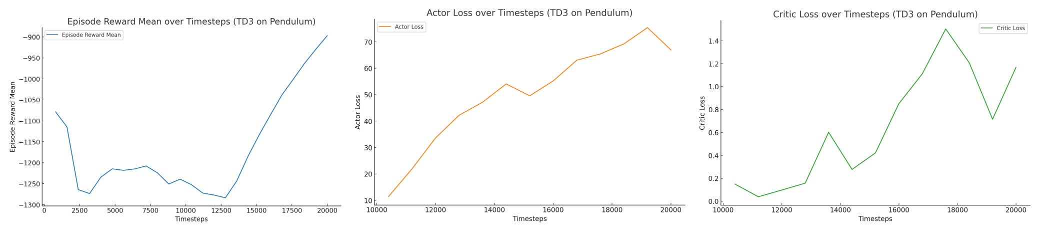
Learning Stability: To evaluate learning stability, we analyze key training metrics for A2C, TD3, and SAC algorithms in the two environments. Stability is assessed by observing the smoothness and convergence of the following metrics: Episode reward mean, policy loss, actor loss, value loss, critic loss, entropy loss, and entropy coefficient.

Pendulum: Figures 3, 4, and 5 show different algorithm specific reward metrics. A2C shows gradual improvement in rewards but with higher variance, reflecting slower convergence due to its on-policy updates. A2C's policy and value losses decrease smoothly with minor instabilities. TD3 achieves faster convergence with a smoother reward progression, benefiting from delayed actor updates and clipped double Q-learning, which stabilize critic loss. Actor loss decreases steadily, indicating consistent policy optimization. SAC demonstrates the most stable and rapid learning, with smooth reward improvement, minimal actor loss oscillations, and efficient critic loss stabilization. Its dynamic entropy coefficient ensures a balanced transition from exploration to exploitation, contributing to stability. Overall, SAC outperforms A2C and TD3 in terms of stability, highlighting the benefits of entropy-regularized off-policy methods.



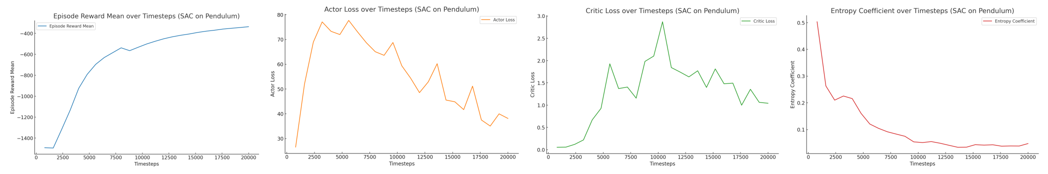
(a) Episode Reward Mean (b) Entropy Loss (c) Policy Loss (d) Value Loss

Figure 3: Learning Stability metrics for A2C on Pendulum environment.



(a) Episode Reward Mean (b) Actor Loss (c) Critic Loss

Figure 4: Learning Stability metrics for TD3 on Pendulum environment.



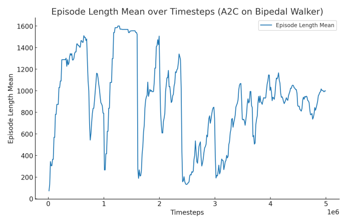
(a) Episode Reward Mean (b) Actor Loss (c) Critic Loss (d) Entropy Coefficient

Figure 5: Learning Stability metrics for SAC on Pendulum environment.

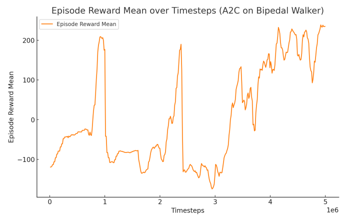
BipedalWalker: Figures 6, 7, and 8 show the learning stability metrics for A2C, TD3, and SAC on the more complex BipedalWalker environment. Similar to the Pendulum environment, A2C shows noisy and inconsistent reward progression, with significant fluctuations in episode length and entropy loss. Similar to A2C in the Pendulum environment, the method's policy and value losses display similar patterns of instability, potentially due to difficulties in optimizing the stochastic on-policy updates. The off-policy methods demonstrate smoother reward growth and stable critic loss. Again, SAC proves to have the best performance in terms of stability.

4.2.2 Evaluation

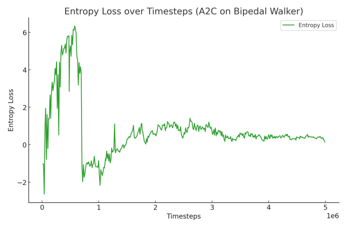
Table 4 shows the mean rewards across 100 evaluation episodes for A2C, TD3, and SAC on the Pendulum and BipedalWalker environments. For the Pendulum environment, TD3 achieves the highest mean reward of -137.95, indicating better performance compared to SAC and A2C. The lower variance in TD3 and SAC suggests more stable performance, whereas A2C demonstrates higher variability, likely due to its on-policy nature.



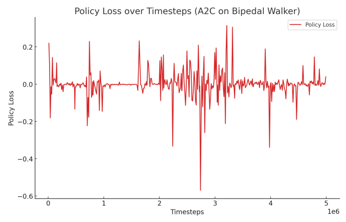
(a) Episode Length Mean



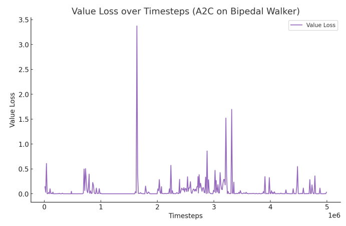
(b) Episode Reward Mean



(c) Entropy Loss

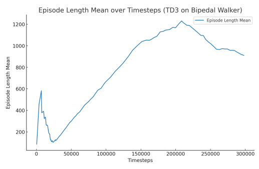


(d) Policy Loss

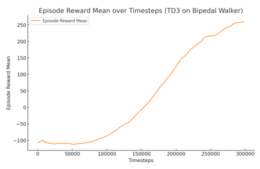


(e) Value Loss

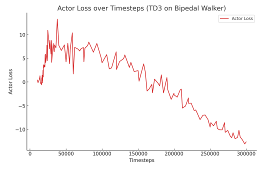
Figure 6: Learning Stability metrics for A2C on Bipedal Walker.



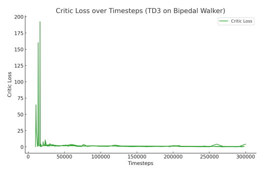
(a) Episode Length Mean



(b) Episode Reward Mean

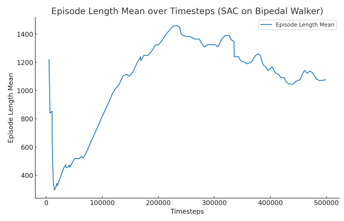


(c) Actor Loss

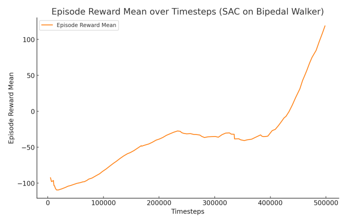


(d) Critic Loss

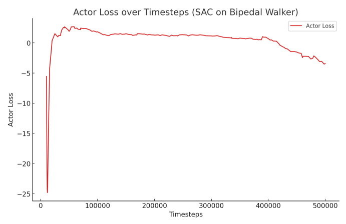
Figure 7: Learning Stability metrics for TD3 on Bipedal Walker.



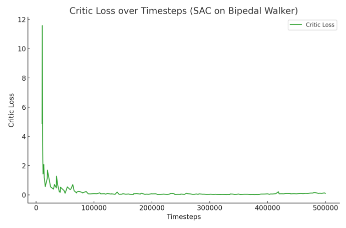
(a) Episode Length



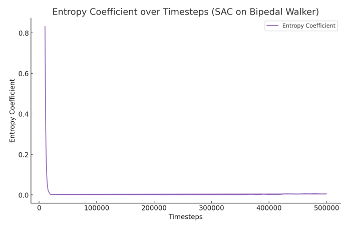
(b) Episode Reward



(c) Actor Loss



(d) Critic Loss



(e) Entropy Coefficient

Figure 8: Learning Stability metrics for SAC on Bipedal Walker.

For the BipedalWalker environment, SAC outperforms the other algorithms with a mean reward of 298.60, highlighting its ability to learn robust policies efficiently. TD3 follows closely with 282.66, showing strong performance but higher variance. TD3's high variance could be attributed to its reliance on deterministic policy gradients. It makes sense that this weakness appeared in the more complex environment of BipedalWalker over the simpler Pendulum. Not surprisingly, A2C lags behind the off-policy methods. These results confirm SAC's stability and sample efficiency across environments and highlight TD3's potential weakness in more complex environments.

Table 4: Mean reward across 100 evaluation episodes for A2C, TD3, and SAC on Pendulum and BipedalWalker environments. Where Pendulum has a maximum reward of 0 and BipedalWalker is considered solved with a reward of 300.

Algorithm	**Pendulum**	**BipedalWalker**
A2C	-179.62 ± 113.21	257.01 ± 49.32
TD3	-137.95 ± 76.52	282.66 ± 75.55
SAC	-157.21 ± 84.54	298.60 ± 35.87

Robustness: To evaluate robustness, we tested A2C, TD3, and SAC across multiple random seeds in the Pendulum and BipedalWalker environments (Table 5 and Table 6).

In the Pendulum environment, the trend across seeds for each algorithm remains consistent. A2C displays high variability across seeds, while TD3 and SAC exhibit more stable performance.

In the more complex BipedalWalker environment, the on-policy algorithm A2C is more sensitive to seed variability. This is demonstrated by the inconsistent performance, ranging from a negative mean reward of -76.21 with seed 2024 to a strong performance of 299.15 with seed 42. This highlights the method's dependence on samples during training. This result is intuitive, as on-policy methods generally require more data to learn effectively and rely heavily on the quality of each sampled trajectory. In contrast, off-policy algorithms like TD3 and SAC exhibit greater robustness across seeds due to their ability to reuse past experiences and perform more stable updates. SAC demonstrates the most robust performance, maintaining consistent high mean rewards across all seeds. This stability can be attributed to SAC's entropy-regularized stochastic policy, which aids in balancing exploration and exploitation.

Overall, the results emphasize that off-policy algorithms (TD3 and SAC) are more robust to variations in seeds compared to the on-policy A2C algorithm.

Table 5: Robustness of A2C, TD3, and SAC on Pendulum across different seeds.

Algorithm	**Seed**	**Mean Reward**
A2C	0	-197.26 ± 112.30
	42	-203.66 ± 103.61
	2024	-221.99 ± 170.41
TD3	0	-154.32 ± 76.10
	42	-145.68 ± 86.28
	2024	-131.87 ± 77.71
SAC	0	-136.50 ± 94.12
	42	-149.61 ± 78.26
	2024	-144.45 ± 78.10

5 Discussion and Conclusion

5.1 Summary of Results

This report investigated the performance of A2C, TD3, and SAC algorithms across two continuous control environments; Pendulum and BipedalWalker. The results highlight differences in both training and testing performance, stability, and robustness between on-policy and off-policy methods.

Table 6: Robustness of A2C, TD3, and SAC on BipedalWalker across different seeds.

Algorithm	Seed	Mean Reward
A2C	0	8.61 ± 4.18
	42	299.15 ± 30.15
	2024	-76.41 ± 22.98
TD3	0	258.80 ± 104.20
	42	300.60 ± 0.72
	2024	270.48 ± 94.06
SAC	0	297.93 ± 32.34
	42	291.05 ± 64.85
	2024	303.74 ± 1.04

During training, SAC demonstrated faster and smoother reward improvement in both environments due to the method's features like entropy-regularized updates. This feature along with it being an off-policy method that utilizes past experiences allows the algorithm to balance exploration and exploitation with better sample efficiency than on-policy methods. TD3 also showed strong training performance with stable improvement but with higher variance in the more complex environment. The on-policy method A2C showed the slowest training progression with high variance, especially in BipedalWalker. These results draw attention to the challenges of on-policy methods in sample efficiency and stability.

In terms of evaluation performance, TD3 achieved the highest mean reward in Pendulum. Features of the algorithm like delayed updates and clipped double Q-learning assist in the method's ability to stabilize value estimation. In BipedalWalker, SAC outperformed all algorithms in terms of mean reward, smallest variance, and consistency across different seeds. This shows its ability to learn in complex environments. A2C underperformed in both environments, particularly in terms of seed sensitivity.

These findings underscore that off-policy methods (TD3 and SAC) are better suited for complex environments that require sample efficiency and robust performance, while on-policy methods like A2C can work for simpler tasks with proper seed selection. This aligns with the existing literature, where SAC is regarded as a high-performing algorithm in environments that require efficient exploration and robust learning. The observed weakness of A2C has also been well documented in the domain, especially in complex tasks where on-policy methods are less sample efficient.

5.2 Conclusion

While this study explored the strengths and weaknesses of on-policy and off-policy methods in continuous control tasks, future work could explore several directions. First, evaluating these algorithms on higher-dimensional environments such as PyBullet environments (Ellenberger [2018–2019]). Second, expanding this analysis to more sophisticated, current state-of-the-art methods rather than looking at established baselines. Finally, expanding the analysis to testing popular RL algorithms in real-world continuous control problems such as autonomous driving or robotic locomotion, where robustness and sample efficiency are critical.

Overall, this study highlights the need for robust, sample-efficient algorithms like SAC and TD3 for solving complex continuous control problems while identifying areas where on-policy methods like A2C can still serve as a reliable baseline.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Bernd Bischl, Martin Binder, Michel Lang, Triin Pärnamaa, and Florian Pfisterer. Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. *Journal of Machine Learning Research*, 22(1):1–45, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL `https://arxiv.org/abs/1606.01540`.

Erin Catto. Box2d: A 2d physics engine for games, 2011. URL `https://box2d.org`. Accessed: [your access date].

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning, 2019. URL `https://arxiv.org/abs/1904.12901`.

Benjamin Ellenberger. Pybullet gymperium. `https://github.com/benelot/pybullet-gym`, 2018–2019.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures, 2018. URL `https://arxiv.org/abs/1802.01561`.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, pages 1587–1596, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019. URL `https://arxiv.org/abs/1509.02971`.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 1928–1937, 2016.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. Ray: A distributed framework for emerging ai applications, 2018. URL `https://arxiv.org/abs/1712.05889`.

OpenAI. Openai gym, 2016. URL `https://gym.openai.com/`.

Jack Parker-Holder, Minqi Jiang, Zheng Xu, Kimin Lee, Edward Grefenstette, Peter Stone, and Shimon Whiteson Sun. Automated reinforcement learning (autorl): A survey and open problems. *Journal of Artificial Intelligence Research*, 73:819–865, 2022.

Antonin Raffin. Rl baselines3 zoo. `https://github.com/DLR-RM/rl-baselines3-zoo`, 2020.

Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations, 2021. URL `https://github.com/DLR-RM/stable-baselines3`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2nd edition, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL `https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf`.

J. K. Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis Santos, Rodrigo Perez, Caroline Horsch, Clemens Dieffendahl, Niall L. Williams, Yashas Lokesh, and Praveen Ravi. Pettingzoo: Gym for multi-agent reinforcement learning, 2021. URL `https://arxiv.org/abs/2009.14471`.

Qing Wang, Jiechao Xiong, Lei Han, peng sun, Han Liu, and Tong Zhang. Exponentially weighted imitation learning for batched historical data. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/4aec1b3435c52abbdf8334ea0e7141e0-Paper.pdf`.

Ronald J. Williams. Reinforcement-learning connectionist systems. Technical Report NU-CCS-87-3, College of Computer Science, Northeastern University, Boston, MA, 1987.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992. ISSN 1573-0565. doi: 10.1007/BF00992696. URL `https://doi.org/10.1007/BF00992696`.

Yuhuai Wu, Elman Mansimov, Shun Liao, Roger Grosse, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation, 2017. URL `https://arxiv.org/abs/1708.05144`.