Original Research Article

How do Different Reinforcement Learning Optimization Models Perform in Robotic Tasks?

Wenjin Zhao[1], Sara Pohland[2]

[1]The High School Affiliated to Renmin University of China, 37 Zhongguancun Street, Beijing, Beijing, 100080, China; [2]University of California, Berkeley, 2594 Hearst Ave, Berkeley, CA 94720, United States

ABSTRACT

Reinforcement Learning (RL) has proven to be a powerful and versatile framework for solving complex problems. It has demonstrated success in areas ranging from game theory to robotic control to autonomous navigation and path planning. However, the rapid development of numerous RL algorithms has outpaced the field's ability to provide clear, standardized comparisons, leaving practitioners with limited guidance for selecting the most appropriate algorithm for a given task. This work addresses this gap by conducting a systematic empirical study comparing five prominent RL algorithms–Vanilla Policy Gradient (VPG), Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Soft Actor-Critic (SAC)–across three distinct task categories: locomotion (Humanoid), continuous control (LunarLander), and navigation (FrozenLake). To ensure a fair comparison, we first performed a hyperparameter sweep for each algorithm in each environment. The final evaluation, which is based on average return, sample efficiency, and training stability, reveals that no single algorithm dominates in all domains. The key findings are that SAC is the superior choice for complex, continuous control, achieving the highest average episode return in both Humanoid and LunarLander; VPG performs surprisingly well in discrete, sparse-reward settings, achieving the highest average episode return in FrozenLake; and a critical trade-off exists between peak performance and training stability. Our findings aid practitioners in understanding the trade-offs to be considered in RL algorithm selection for different types of robotics tasks.

Keywords: Reinforcement learning; Comparison; Robotic control; Mujoco; Spinning Up

INTRODUCTION

Reinforcement Learning (RL) has emerged as a powerful framework, inspired by behavioral psychology. It involves an agent that learns to achieve a goal through iterative interactions with the environment. The agent receives rewards or penalties for its actions and discovers a policy–a mapping from states to actions–that maximizes its total reward over time. While the theoretical foundations of RL have been established for decades, the applications of RL have been limited to domains with small discrete state and action spaces (1, 2). The emergence of Deep Reinforcement Learning (DRL) marked a shift in overcoming this fundamental barrier. By integrating

Corresponding author: Wenjin Zhao, E-mail: Williamzhaowj@outlook.com.

Accepted December 8, 2025
https://doi.org/10.70251/HYJR2348.36914922

deep neural networks, DRL algorithms could learn to generalize from past experiences to novel states (3).

The improvements in DRL have shown profound potential and have already demonstrated success across a diverse range of fields, which can be categorized into several key areas. First, it demonstrated its strong capabilities in game theory and strategic plays. Systems like Alpha Go and its successor, Alpha Zero, demonstrated RL's capabilities by achieving superhuman performance in complex games like GO and chess without relying on human data (4). This approach has established self-play RL as a powerful method for mastering competitive environments with perfect information (4). Also, their abilities could be applied to robotics and control systems. Multiple studies were done focused on using RL for sophisticated robotic tasks. This includes developing controllers for legged robots to achieve stable locomotion over terrains (5), as well as training robotic arms for precise grasping and assembling operations (6, 7). Consequently, RL has seen expanding use in autonomous navigation and path planning. RL has been applied to navigation problems, from mobile robot path planning in dynamic environments (8) to more complex scenarios involving multi-robot systems (8). These tasks tackle the challenges of obstacle avoidance and efficient exploration.

Despite these advancements, a clear gap remains. Major reviews consistently highlight Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) as top-performing model-free deep RL algorithms for robotic control (8). However, these surveys often conclude their results from studies that evaluate algorithms under disparate conditions, metrics, and environmental setups (9). This leads to various limitations. First, it is difficult to make direct, fair comparisons between different RL algorithms because results are aggregated among different studies conducted under inconsistent experimental conditions (10, 11). Differences in critical factors such as hyperparameter tuning strategies, reward function, and even minor version differences in the simulation environment might significantly impact the performance, obscuring the true relative advantage of each algorithm (10). Second, many evaluations prioritize a single metric, such as the final task performance. This overlooks crucial aspects of practical deployment, such as sample efficiency, training stability, and the requirements for computational power (6, 10). Third, though algorithms such as PPO are noted for continuous control (1, 8), and SAC for high-dimensional spaces (1), a systematic evaluation of their performance across distinct task categories under the same experimental framework is lacking (9). This makes it challenging to select the most suitable algorithm for a specific task type.

Our research addresses this lack of algorithm comparison by conducting a systematic study on how different RL algorithms perform in robotics tasks, including locomotion, continuous control, and navigation, under standardized conditions. In particular, we make the following contributions. First, we adopted a standardized benchmarking system using OpenAI's Spinning Up framework (12) and the MuJoCo physics simulator (13). This ensures that all algorithms are trained and evaluated on the same benchmark tasks with identical conditions and random seeds. Second, we made multi-dimensional performance evaluations. We moved beyond a single metric and evaluated the algorithm's performance across a comprehensive set of criteria. Finally, we made sure of a fair comparison among algorithms. We conducted a structured hyperparameter optimization process for each algorithm-task pair. This allows us to analyze the performance of a well-tuned algorithm, rather than a default configuration. Overall, this study aims to provide a standardized and comprehensive evaluation of state-of-the-art RL algorithms across multiple robotics task categories, offering practitioners a guide for selecting the most suitable algorithm for their application.

METHODS AND MATERIALS

Reinforcement Learning (RL) Algorithms

For this comparative study, we selected five prominent deep reinforcement learning algorithms: Vanilla Policy Gradient (VPG) (14), Deep Deterministic Policy Gradient (DDPG) (15), Proximal Policy Optimization (PPO) (16), Twin Delayed DDPG (TD3) (17), and Soft Actor-Critic (SAC) (18). This selection of algorithms covers foundational and state-of-the-art methods, representing both on-policy and off-policy approaches (11), as well as both stochastic and deterministic policies. This selection allows for a more comprehensive comparison and analysis of algorithm strengths and weaknesses across different environments.

The earliest of these methods, VPG, represents a classic on-policy algorithm that directly optimizes the policy by ascending the gradient of the expected reward. Building upon the need to handle continuous action spaces more efficiently, DDPG was introduced as an off-policy actor-critic algorithm designed for environments with continuous action spaces. To counteract the limitations of standard policy gradient methods, PPO later emerged as an improved on-policy alternative by using a clipped objective function, enhancing training stability. As research progressed, TD3 was introduced as an evolution of DDPG, refining the off-policy actor-critic structure by addressing the overestimation bias issues found in its predecessor. Finally, SAC emerges as an off-policy actor-critic architecture that integrates entropy regularization, encouraging more exploration and enhancing robustness during training.

Simulation Environment & Tasks

A critical step for a comparative analysis of RL algorithms is the selection of appropriate benchmarking environments. For this study, we selected three distinct environments: Humanoid (19), Lunarlander (20), and FrozenLake (21) from the Gym environments. This selection is made to ensure coverage across the three fundamental categories of RL problems: locomotion, control, and navigation tasks.

Humanoid

The humanoid environment represents the class of high-dimensional, continuous control problems, focused on locomotion. The agent's objective is to make a 3D humanoid model walk forward without falling. Having 376-dimensional state observations and 17 actuators, the humanoid environment requires the algorithm to learn sophisticated coordination and balance.

LunarLander

The goal in the LunarLander environment is to pilot a lunar module to land safely on a designated pad. This environment is a more tractable but non-trivial control problem with an 8-dimensional state space and a 2-dimensional continuous action space, testing the algorithm's ability to deal with continuous dynamics and precise throttle control.

FrozenLake

The FrozenLake environment represents a discrete, sparse-reward navigation and planning problem. The agent's objective is to navigate from a start tile to a goal on an 8x8 grid without falling into holes. With a 64-state observation space and 4 discrete actions (up, down, left, right), this environment demonstrates the algorithm's ability to learn optimal paths under sparse reward conditions.

Policy Hyperparameters

To ensure a fair and rigorous comparison, we conducted a tuning process for each algorithm in each environment. The goal is to isolate the difference between algorithms, rather than compare their default configurations. We selected discount factor (gamma)–a hyperparameter between 0 to 1 that determines the importance of future rewards compared to immediate rewards–and learning rate, which controls the steps taken to update the parameters of the agent's policy. Tuning these hyperparameters balances training speed and stability, as they are common across all algorithms and have the most prominent effect on the learned policy.

As for the optimization process, we trained all of the policies for 100 iterations across five distinct values for each key hyperparameter. This method isolated the impact of one individual hyperparameter while maintaining consistency across other conditions. The training progress was logged at each epoch, capturing important metrics, including average episode return, policy and value function losses, and Q-value (for critic-based algorithms). The better-performing hyperparameter would be chosen based on these metrics and would be used for comparison across algorithms for a given task.

Comparison across Algorithms

We evaluated each algorithm based on three metrics. First, we compared the average episode return. We recorded the cumulative reward obtained in each episode throughout training. The final performance would be evaluated based on the maximum reward achieved during training and the value it converged to. Second, we considered sample efficiency. We measured the speed of learning by calculating the number of environment interactions required for an algorithm to achieve a performance threshold. This allows us to compare how quickly each algorithm learns a high-performance policy. Third, we evaluated training stability, using the standard deviation of the episode return across epochs. A lower variance indicates a more stable and predictable training progression, which is crucial for practical applications because high instability complicates the identification of a performant final policy and poses significant safety risks in real-world deployment, where unpredictable behaviors can cause hardware damage.

To better facilitate and analyze the learning trend and reduce the visual noise in the raw training data, we applied a low-pass filter to the episode return curves. This process smoothes out short-term, high-frequency fluctuations, while preserving the long-term trends. This makes the comparison of the learning trajectory more interpretable and more straightforward.

RESULTS & DISCUSSION

Comparison across Hyperparameter Values

After analyzing the final results, we selected the best hyperparameter values to be used for the final comparison, which are provided in Table 1.

For all remaining hyperparameters, we kept the standard default settings of the algorithms. All methods had the same rollout and network settings: the algorithms ran with 4,000 environment steps per epoch, a neural network with two hidden layers with 256 units each, and a maximum episode length of 1,000 steps. We took a uniform configuration for the off-policy algorithms (DDPG, TD3, and SAC), including a mini-batch size of 100 samples, a 10,000 random steps pre-learning warm-up phase, and update rules that implemented a gradient update after the first 100 steps of experience collection and performed the update every 50 steps thereafter. DDPG and TD3 added exploration noise with an intensity of 0.1 applied to the actions. TD3 also used its signature regularization elements: target policy smoothing by using noise with a standard deviation of 0.2, limiting noise clipping to an absolute value of 0.5, and a delayed policy update, which updated the policy network only once every two critic updates. SAC used entropy-temperature only with a coefficient set to 0.2. The on-policy algorithms adopted a generalized advantage estimation (GAE) decay factor of 0.97 for both PPO and VPG. PPO additionally utilized a clipping threshold of 0.2 for its surrogate objective, implemented 80 optimization iterations on the policy per update cycle, and controlled a target KL-divergence of 0.01 to manage training stability. By contrast, VPG performed 80 iterations of only its value function updates.

Table 1. Hyperparameters used for every Environment-Algorithm pair in the final comparison.

Environment	Algorithm	Gamma	Learning Rate
Humanoid	PPO	0.91	pi_lr :0.0030 vf_lr: 0.0003
Humanoid	DDPG	0.93	pi_lr: 0.0001 q_lr: 0.0003
Humanoid	SAC	0.99	lr: 0.0010
Humanoid	TD3	0.99	pi_lr: 0.0001 q_lr: 0.0030
Humanoid	VPG	0.99	pi_lr: 0.0100 vf_lr: 0.0100
LunarLander	PPO	0.97	pi_lr: 0.0030 vf_lr: 0.0100
LunarLander	DDPG	0.99	pi_lr: 0.0010 q_lr: 0.0003
LunarLander	SAC	0.99	lr: 0.0010
LunarLander	TD3	0.99	pi_lr: 0.0100 q_lr: 0.0030
LunarLander	VPG	0.99	pi_lr: 0.0030 vf_lr: 0.0100
Frozenlake	PPO	0.99	pi_lr: 0.0001 vf_lr: 0.0010
Frozenlake	DDPG	0.95	pi_lr: 0.0100 q_lr: 0.0010
Frozenlake	SAC	0.91	lr: 0.0100
Frozenlake	TD3	0.93	pi_lr: 0.0010 q_lr: 0.0010
Frozenlake	VPG	0.99	pi_lr: 0.0100 vf_lr: 0.0010

Comparison of RL Algorithms

Humanoid

As shown in Figure 1, for Average Episode Return, **SAC** achieved the highest performance, with a maximum average episode return of approximately 5,100 and converged to a value around 4,800. **TD3** was the second-best performer, reaching a maximum of 5,000 (about 2% less than SAC's peak) and stabilizing around 4,000 (about 16.7% less than SAC's value after convergence). The remaining algorithms–**DDPG**, **PPO**, and **VPG**–performed significantly worse, with final returns of 536, 318, and 199, respectively (about 11%, 6.6%, and 4.17% of SAC's value after convergence). These lower returns can be partially attributed to their algorithmic limitations, as **PPO** and **VPG** suffer from high variance in gradient estimates in high-dimensional continuous control tasks, and **DDPG** is sensitive to exploration noise.

As for Sample Efficiency, **SAC** demonstrated superior sample efficiency, converging to its high-performance plateau by approximately epoch 75. In contrast, **TD3** required around 225 epochs to reach its converged value (three times slower than SAC). While **DDPG**, **PPO**, and **VPG** achieved their low performance levels within 50 epochs, their poor performance made comparing sample efficiency less relevant compared to higher-performing algorithms.

Finally, for Training Stability, **PPO** and **VPG** were the most stable after convergence with very low standard deviations (σ = 6.4 and 5.0 from epoch 0 to

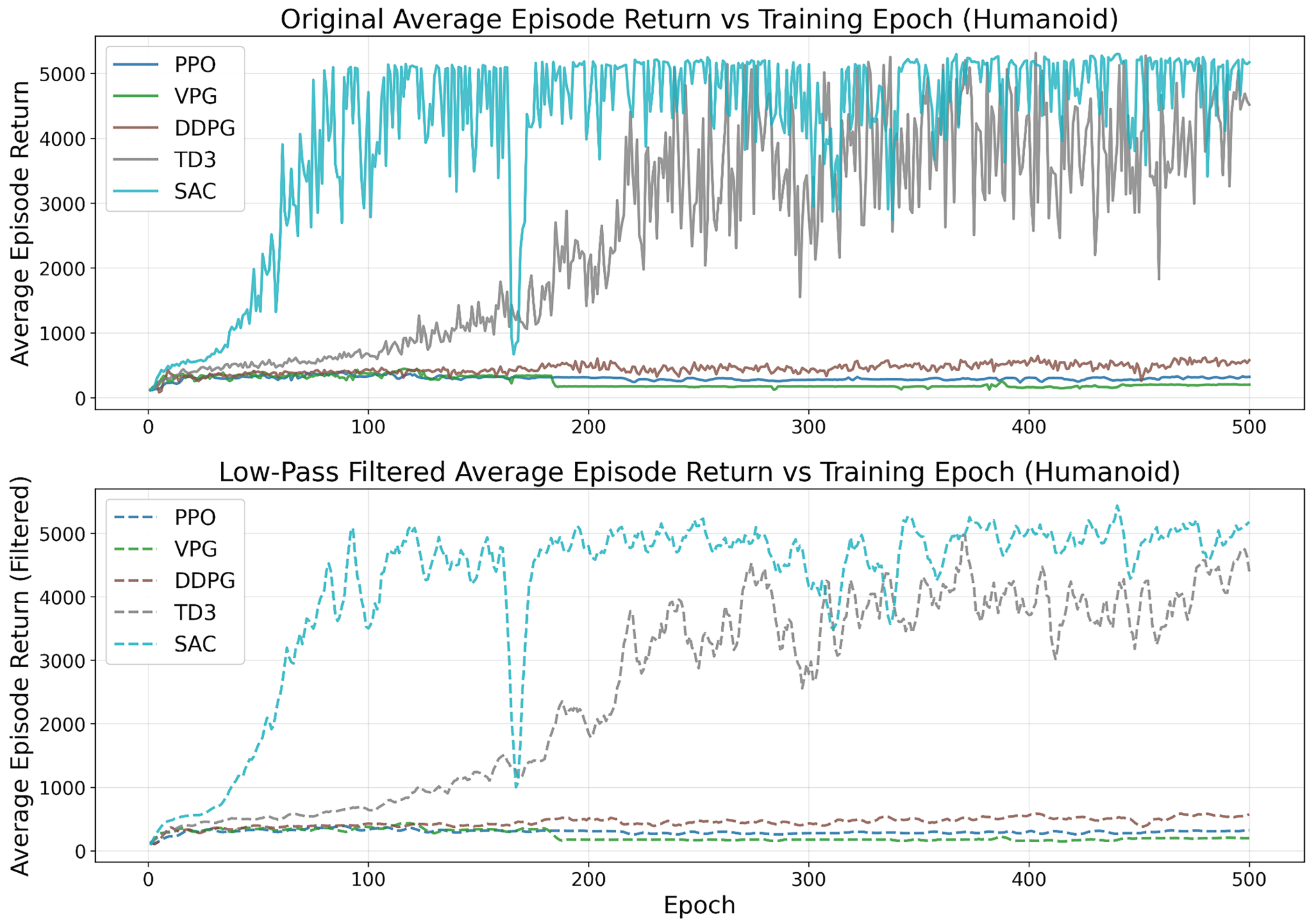

Figure 1. Learning curves comparing average episode return across five RL algorithms for the Humanoid environment.

500). **DDPG** was moderately stable (σ = 31 from epoch 0 to 500). However, the top performers, **SAC** and **TD3**, exhibited high variance (σ = 641 from epoch 100 to 500; and 513 from epoch 200 to 500), indicating that their high-performance training, while successful, is less predictable.

LunarLander

As shown in Figure 2, for Average Episode Return, SAC again delivered the best performance, with a best return of 284.7 and a stable average of 280 after convergence. DDPG was a close second, achieving a best return of 266.9 (6.25% less than SAC's peak) and an average of 260 (7.2% less than SAC's value after convergence). TD3 failed to converge effectively, plateauing at a much lower average of 100, while VPG and PPO performed poorly. The poor performance of VPG and PPO is due to their on-policy nature and high-variance gradient updates, which make them unsuitable for LunarLander. TD3 fails largely because it was designed for continuous action spaces and relies on values with lesser noise, making it not suitable for LunarLander.

As for Sample Efficiency, SAC was the most efficient, reaching a return of 200 by epoch 20 and converging by epoch 100. DDPG was less efficient, requiring until epoch 80 to reach a return of 225. TD3, VPG, and PPO showed minimal learning progress after the first 20 epochs.

Finally, for Training Stability, SAC was exceptionally

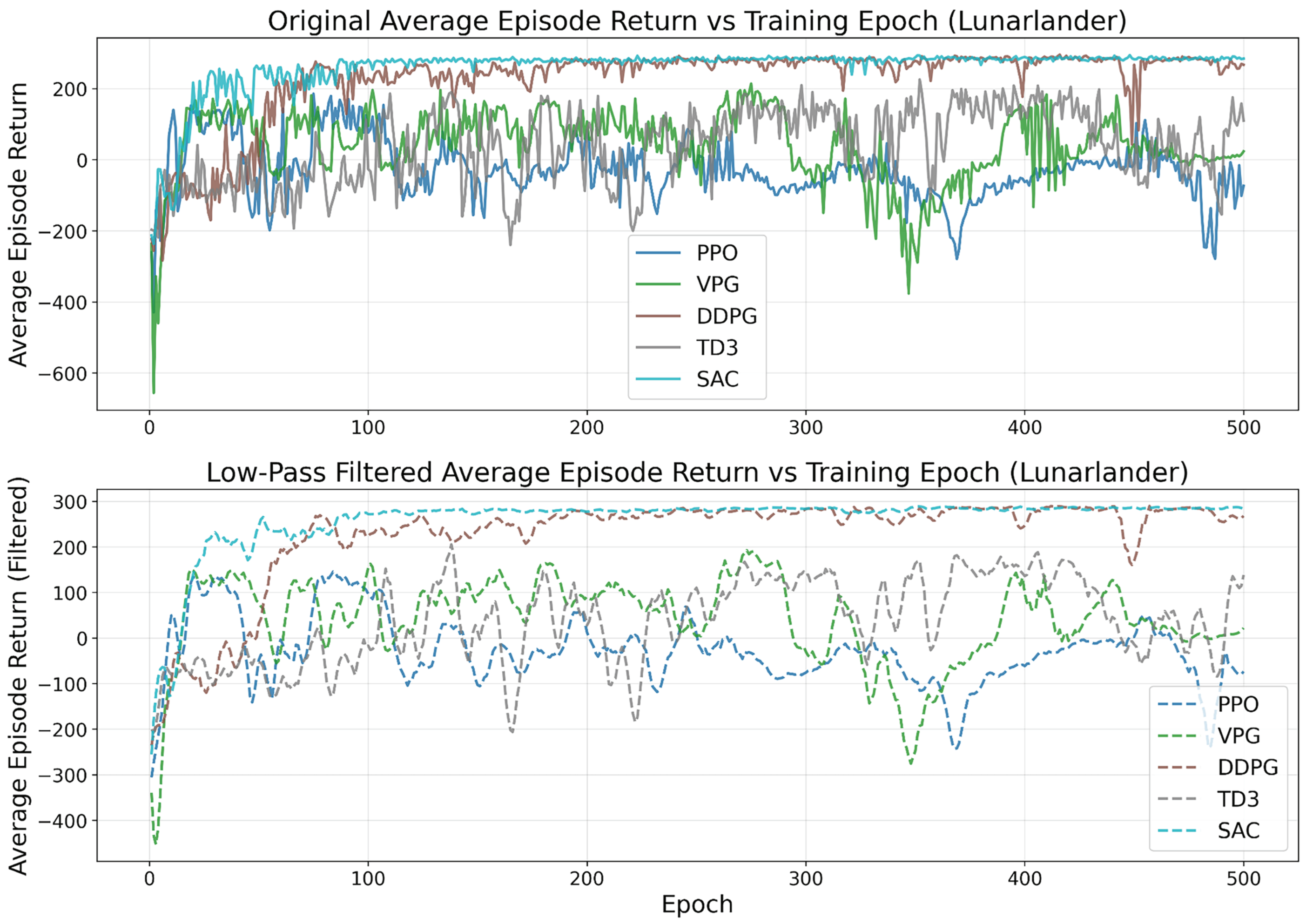

Figure 2. Learning curves comparing average episode return across five RL algorithms for the LunarLander environment.

stable (σ = 5.43 from epoch 50 to 500), reinforcing its robustness. DDPG was less stable (σ = 15.29 from epoch 50 to 500), and TD3, VPG, and PPO showed high instability (σ = 30.11, 39.47, and 43.69 from epoch 50 to 500).

<u>FrozenLake</u>

As shown in Figure 3, for Average Episode Return, VPG was the standout algorithm, consistently solving the maze to achieve the maximum success rate of 1.0. DDPG was the only other algorithm to show significant progress in learning a good RL policy, reaching a maximum success rate of 0.55 (average 0.3). TD3 achieved a modest peak of 0.3 before declining, while PPO and SAC failed to learn, remaining near zero throughout training. DDPG's off-policy nature allows its moderate learning despite its weak exploration, causing it to have a relatively lower performance. PPO and SAC fail due to their reliance on stable, incremental policy updates and entropy-driven exploration.

As for Learning Efficiency, VPG learned the optimal policy most efficiently, converging within 50 epochs. DDPG converged by epoch 100, and TD3 by epoch 185. The failure of PPO and SAC to learn meaningful behavior precludes an efficiency analysis for this task.

Finally, for Training Stability, the successfully converging algorithms were highly stable. VPG was the most stable (σ = 0.04 from epoch 100 to 500), followed by DDPG (σ = 0.078 from epoch 100 to 500) and TD3 (σ = 0.117 from epoch 100 to 500).

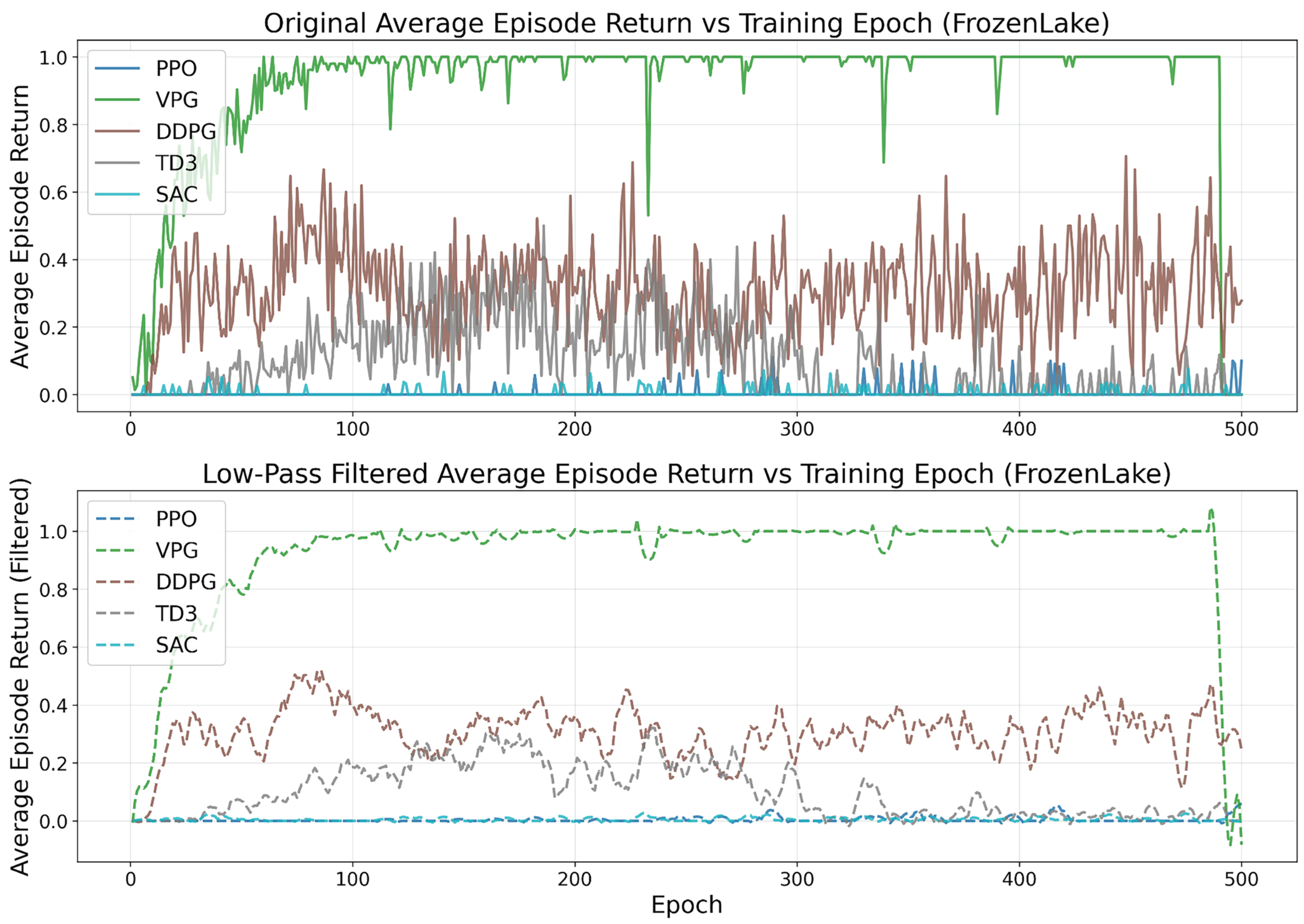

Figure 3. Learning curves comparing average episode return across five RL algorithms for the FrozenLake environment.

CONCLUSION

The results show that an algorithm's performance is not absolute but intrinsically linked to how its mechanics align with the specific demands of a task. The finding can be summarized as follows.

First, the high performance of Off-Policy, Maximum Entropy Learning for Complex Controls: SAC showed consistent high performance in dense-reward, high-dimensional environments like Humanoid and high efficiency in LunarLander. These can be attributed to its off-policy nature and maximum entropy objective. Its ability to reuse experience improves sample efficiency, and its entropy maximization encourages broader exploration of the action space. These are crucial for mastering complex motor skills. This makes SAC the generally recommended algorithm for challenging continuous control problems.

Second, the Specialization of Simple Policy Gradients in Discrete, Sparse Reward Settings: The dominance of VPG in the FrozenLake environment showed a crucial insight: in discrete, sparse-reward environments, excessive optimization can be detrimental. While advanced methods like SAC and PPO rely on sophisticated value estimates and policy constraints, they could converge on exploitation at an early stage. On the other hand, VPG's simpler, more randomly determined policy updates, though less efficient, are more resistant to this pitfall. This allowed it to effectively reinforce the successful trajectories in a simple grid world.

Third, the Balance Between Performance and Stability: There's a clear trade-off between peak performance and training stability. Top performers in continuous control, SAC, and TD3, though achieving the highest rewards, showed a significant variance between epochs. This is a known characteristic of their complex, off-policy learning dynamics. In contrast, PPO, with its on-policy updates and clipped objective, which served as a stabilizer, ensured a more consistent but often more mediocre result. This positions PPO as a reliable but less ambitious choice for applications where predictability is prioritized over peak performance.

Finally, the Critical Role of the Action Space: The results differentiate algorithms' suitability based on the action spaces. In continuous action spaces (Humanoid, LunarLander), actor-critic methods like SAC, TD3, and DDPG are necessary. However, in discrete action spaces (FrozenLake), simpler methods like VPG unexpectedly showed higher performance, due to the reduced complexity of the policy.

In summary, the "best" algorithm depends on the problem context. For dense-reward, continuous control, modern off-policy algorithms like **SAC** are superior. For discrete, sparse-reward navigation, the problem's simplicity can make advanced methods ineffective, making simpler algorithms like **VPG** the most suitable. For practitioners, it's important to match the algorithm's characteristics–its handling of action space, exploration strategy, and stability mechanisms–to the fundamental challenge of the tasks.

While this study provides a comprehensive comparison across fundamental categories, it also shows promising directions for future work. During our research, we experimented with environments like FetchReach and CarRacing and observed a notable decrease in performance across all algorithms. We hypothesize that these tasks may be better suited to different paradigms. Future work should compare algorithms against methods that incorporate imitation learning.

Also, this study was conducted completely in simulation. A critical next step is to validate these findings through real-world robotic deployment. The performance we observed may shift when algorithms are confronted with real-world challenges like sensor noise, actuator delay, and environmental stochasticity. Therefore, investigating the sim-to-real transfer capabilities of these algorithms is an essential area for future work to bridge the gap between benchmark performance and practical utility.

CONFLICT OF INTEREST

The author declares that there are no conflicts of interest related to this work.

REFERENCES

1. AlMahamid F, Grolinger K. Reinforcement Learning Algorithms: An Overview and Classification. *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE).* 2021 Sep 12; 1-7. https://doi.org/10.1109/CCECE53047.2021.9569056
2. Akalin N, Loutfi A. Reinforcement Learning Approaches in Social Robotics. *Sensors*. 2021 Feb 11; 21 (4): 1292. https://doi.org/10.3390/s21041292
3. Shaheen A, Badr A, Abohendy A, Alsaadawy H, Alsayad N. Reinforcement Learning in Strategy-Based and Atari Games: a Review of Google DeepMinds Innovations.arXiv:2502.10303 [Preprint]. 2025 Feb 22. Available from: https://arxiv.org/abs/2502.10303 (accessed on 2025-11-18)
4. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, *et al*. Mastering the game of Go without human knowledge. *Nature*. 2017 Oct; 550 (7676): 354-9. https://doi.org/10.1038/nature24270
5. MdAM Khan, MRJ Khan, Tooshil A, Sikder N, *et al*. A Systematic Review on Reinforcement Learning-Based Robotics Within the Last Decade. *IEEE Access*. 2020; 8: 176598-623. https://doi.org/10.1109/ACCESS.2020.3027152
6. Nguyen H, La H. Review of Deep Reinforcement Learning for Robot Manipulation. *2019 Third IEEE International Conference on Robotic Computing (IRC).* 2019 Feb; 590-5. https://doi.org/10.1109/IRC.2019.00120
7. Bhagat S, Banerjee H, Ho Tse ZT, Ren H. Deep Reinforcement Learning for Soft, Flexible Robots: Brief Review with Impending Challenges. *Robotics*. 2019 Jan 18; 8 (1): 4. https://doi.org/10.3390/robotics8010004
8. Garaffa LC, Basso M, Konzen AA, de Freitas EP. Reinforcement Learning for Mobile Robotics Exploration: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*. 2023 Aug; 34 (8): 3796-810. https://doi.org/10.1109/TNNLS.2021.3124466
9. Liu R, Nageotte F, Zanne P, de Mathelin M, Dresp-Langley B. Deep Reinforcement Learning for the Control of Robotic Manipulation: A Focussed Mini-Review. *Robotics*. 2021 Jan 24; 10 (1): 22. https://doi.org/10.3390/robotics10010022
10. Henderson P, Islam R, Bachman P, Pineau J, Precup D, Meger D. Deep Reinforcement Learning that Matters. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. 2018 Feb 02; 3207-14. https://doi.org/10.1609/aaai.v32i1.11694
11. Zhang T, Mo H. Reinforcement learning for robot

research: A comprehensive review and open issues. *International Journal of Advanced Robotic Systems*. 2021 May 1; 18 (3). https://doi.org/10.1177/17298814211007305

12. Achiam J. Welcome to Spinning Up in Deep RL! - Spinning Up documentation [Internet]. spinningup.openai.com. 2018. Available from: https://spinningup.openai.com/en/latest/ (accessed on 2025-11-18)
13. DeepMind Technologies Limited. MuJoCo - Advanced Physics Simulation [Internet]. mujoco.org. 2021. Available from: https://mujoco.org/ (accessed on 2025-11-18)
14. Williams RJ. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*. 1992 May; 8 (3-4): 229-56. https://doi.org/10.1023/A:1022672621406
15. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, *et al*. Continuous control with deep reinforcement learning. arXiv:1509.02971 [Preprint]. 2015. Available from: https://arxiv.org/abs/1509.02971 (accessed on 2025-11-18)
16. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [Preprint]. 2017. Available from: https://arxiv.org/abs/1707.06347 (accessed on 2025-11-18)
17. Fujimoto S, Van Hoof H, Meger D. Addressing Function Approximation Error in Actor-Critic Methods [Internet]. 2018 Oct. Available from: https://proceedings.mlr.press/v80/fujimoto18a/fujimoto18a.pdf (accessed on 2025-11-18)
18. Haarnoja T, Zhou A, Abbeel P, Levine S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor [Internet]. Proceedings.mlr.press. PMLR; 2018. p. 1861-70. Available from: https://proceedings.mlr.press/v80/haarnoja18b.html (accessed on 2025-11-18)
19. The Farama Foundation. Gymnasium Documentation [Internet]. gymnasium.farama.org. Available from: https://gymnasium.farama.org/environments/mujoco/humanoid/ (accessed on 2025-11-18)
20. Klimov O. Gymnasium Documentation [Internet]. Farama.org. 2022. Available from: https://gymnasium.farama.org/environments/box2d/lunar_lander/ (accessed on 2025-11-18)
21. The Farama Foundation. Gymnasium Documentation [Internet]. gymnasium.farama.org. Available from: https://gymnasium.farama.org/environments/toy_text/frozen_lake/ (accessed on 2025-11-18)