Preprints.org

Article

Not peer-reviewed version

A Comparative Evaluation of Value-Based and Policy-Gradient Deep Reinforcement Learning Algorithms: Benchmarking on LunarLander-v2 and a Custom Partially Observable Drone-Rescue Environment

Samya Mukherjee * and Sayak Chowdhury

Posted Date: 28 July 2026

doi: 10.20944/preprints202607.2003.v1

Keywords: deep reinforcement learning; Deep Q-Network; Double DQN; REINFORCE; Proximal Policy Optimization; Advantage Actor-Critic; LunarLander; reward shaping; partially observable Markov decision process; autonomous navigation

Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC, OpenAlex.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 July 2026 doi:10.20944/preprints202607.2003.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

A Comparative Evaluation of Value-Based and Policy-Gradient Deep Reinforcement Learning Algorithms: Benchmarking on LunarLander-v2 and a Custom Partially Observable Drone-Rescue Environment

Samya Mukherjee * and Sayak Chowdhury

Department of Data Science, Ramakrishna Mission Vivekananda Educational and Research Institute (RKMVERI), Belur Math, Howrah, West Bengal, India

* Correspondence: samyamukherjee.rs@gm.rkmvu.ac.in

Abstract

Deep Reinforcement Learning (DRL) algorithms are frequently compared on standard benchmark environments, yet the practical implications of algorithmic choice for engineered, partially observable tasks are less often examined side by side with such benchmarks. In this work we present a two-part empirical study. First, we implement five DRL algorithms entirely from scratch in PyTorch – Deep Q-Network (DQN), Double DQN, REINFORCE, Proximal Policy Optimization (PPO), and Advantage Actor-Critic (A2C) – and evaluate them under matched conditions on the LunarLander-v2 control task from the Gymnasium suite. We report learning curves, sample efficiency, stability, and final policy quality, and find that the value-based methods (DQN, Double DQN) solve the environment reliably (in 461 and 798 episodes respectively, reaching average scores above 200), whereas the on-policy methods REINFORCE, PPO, and A2C fail to solve the task within their allotted training budgets despite requiring substantially more environment interactions. Second, we design and evaluate *Snow Drone Rescue*, a custom, partially observable, stochastic environment built in `pygame` that models an autonomous drone performing a multi-objective search-and-rescue mission under battery constraints and adverse weather. A DQN agent trained with a dense-plus-sparse shaped reward learns energy-aware navigation and obstacle avoidance despite receiving only a six-dimensional partial observation. Taken together, the two studies indicate that, for discrete-action tasks with reasonably dense reward signals, algorithmic stability mechanisms (experience replay, target networks) matter more for final performance than the choice between value-based and policy-gradient families, while reward and environment design remain at least as consequential as the learning algorithm itself. We discuss the theoretical basis for these observations, algorithmic and environment-design trade-offs, and directions for future work, including entropy-regularized policy optimization, systematic hyperparameter search, and extension to continuous-control and multi-agent settings.

Keywords: deep reinforcement learning; Deep Q-Network; Double DQN; REINFORCE; Proximal Policy Optimization; Advantage Actor-Critic; LunarLander; reward shaping; partially observable Markov decision process; autonomous navigation

1. Introduction

Deep Reinforcement Learning (DRL) has produced a wide family of algorithms that differ in how they estimate value functions, parameterize policies, and use collected experience. Value-based methods such as Deep Q-Networks (DQN) [1] learn an action-value function and derive a policy implicitly, exploiting off-policy experience replay and a slowly updated target network to stabilize training. Double DQN [2] addresses a specific pathology of DQN – the overestimation of action values

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 July 2026 doi:10.20944/preprints202607.2003.v1

caused by the max operator in the target computation – by decoupling action selection from action evaluation. In contrast, policy-gradient methods such as REINFORCE [3] optimize a parameterized policy directly via Monte Carlo estimates of the policy gradient, and actor-critic variants such as Advantage Actor-Critic (A2C) [5] and Proximal Policy Optimization (PPO) [4] reduce the variance of these estimates using a learned value baseline (the critic) and, in the case of PPO, a clipped surrogate objective that constrains policy updates.

While each of these algorithms has established theoretical guarantees and well-documented empirical behaviour in the literature, most practitioner-facing comparisons are conducted with differing codebases, tuning budgets, and environments, which makes it difficult to isolate the effect of the algorithm itself. A second, related gap is that most publicly reported comparisons are restricted to standard Gymnasium benchmarks, leaving open the question of how these algorithms and, more importantly, the reward-design decisions that accompany them, transfer to engineered tasks with partial observability and multiple competing objectives, which are more representative of applied autonomous-systems settings.

This paper addresses both points through two complementary studies conducted under a single, internally consistent implementation.

1. **A controlled comparative benchmark.** We implement DQN, Double DQN, REINFORCE, PPO, and A2C from scratch in PyTorch and train each on LunarLander-v2, a discrete-action control task with an 8-dimensional continuous state space. Holding the task fixed, we compare the algorithms on sample efficiency, training stability, and asymptotic policy quality, and relate the empirical differences to the algorithms' underlying theoretical properties.
2. **A custom environment case study.** We design *Snow Drone Rescue*, a stochastic, partially observable environment simulating an autonomous drone conducting a search-and-rescue mission over snow-covered terrain under battery constraints. Using the best-performing algorithm identified in the first study (DQN), we examine how reward shaping and partial observability interact to produce (or fail to produce) useful emergent behaviour, such as energy-aware flight paths and implicit obstacle avoidance.

The remainder of the paper is organized as follows. Section 2 presents the formal Markov Decision Process (MDP) formulations for both environments, the network architectures, hyperparameters, and training procedures used throughout. Section 3 reports the empirical results for both studies. Section 4 discusses the theoretical and practical implications of these results, hyperparameter sensitivity, and the limitations of the present work. Section 5 concludes and outlines directions for future research.

2. Materials and Methods

2.1. Study 1: LunarLander-v2 Benchmark

2.1.1. Task and MDP Formulation

LunarLander-v2 (Gymnasium) requires an agent to pilot a lander to a safe touchdown on a marked pad using three thrusters. We formalize the task as an MDP $M = (\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma)$.

State space. The state is an 8-dimensional continuous vector

$$s = [x, y, v_x, v_y, \theta, \omega, c_\ell, c_r]^\top,$$

where (x, y) are the normalized horizontal and vertical positions, (v_x, v_y) the corresponding velocities, θ the angular orientation, ω the angular velocity, and $c_\ell, c_r \in \{0, 1\}$ indicate left- and right-leg ground contact.

Action space. The action space is discrete, $\mathcal{A} = \{0, 1, 2, 3\}$, corresponding to *no-op*, *fire left engine*, *fire main engine*, and *fire right engine*.

Reward function. The reward decomposes as

$$R(s, a, s') = R_{\text{landing}} + R_{\text{crash}} + R_{\text{fuel}} + R_{\text{proximity}} + R_{\text{leg}},$$

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 July 2026 doi:10.20944/preprints202607.2003.v1

with fuel costs of -0.3 for the main engine and -0.03 per side-engine firing, a ± 10 leg-contact shaping term, a proximity-based shaping term that rewards moving toward the pad, and terminal rewards of $+100$ to $+140$ for a safe landing and -100 for a crash.

Discount factor. We use $\gamma = 0.99$, standard for episodic tasks requiring long-horizon credit assignment.

Termination. Episodes terminate on a soft landing, a crash, the lander leaving the viewport, or truncation at 1000 steps.

2.1.2. Algorithms

All five algorithms were implemented from scratch in PyTorch to ensure a controlled, internally consistent comparison; no third-party RL library was used for the core update rules.

DQN.

The Q-network $Q_\theta(s, a)$ is a feedforward network with two 64-unit ReLU hidden layers mapping the 8-dimensional state to four action values. Training minimizes the temporal-difference loss

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\Big[\big(r + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a)\big)^2\Big],$$

using a replay buffer $\mathcal{D}$ of capacity 10^5, a target network Q_{θ^-} updated with soft update rate $\tau = 10^{-3}$, and ϵ-greedy exploration decayed geometrically from 1.0 to 0.01 at rate 0.995 per episode.

Double DQN.

Double DQN uses an identical network and replay mechanism, but decouples action selection from evaluation in the target computation:

$$a^* = \arg\max_{a'} Q_\theta(s', a'), \qquad y = r + \gamma\, Q_{\theta^-}(s', a^*)(1 - d),$$

where d indicates episode termination, mitigating the overestimation bias of vanilla DQN's max operator.

REINFORCE.

A policy network with the same 64-64 hidden architecture outputs action logits; actions are sampled from the induced categorical distribution $\pi_\theta(a \mid s)$. Parameters are updated via the Monte Carlo policy-gradient estimator

$$\nabla_\theta J(\theta) = \mathbb{E}\Big[\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\, G_t\Big],$$

with G_t the discounted return from time t, and no learned baseline.

PPO.

PPO uses separate actor and critic networks (each 64-64 ReLU) and optimizes the clipped surrogate objective

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\Big], \qquad r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)},$$

with advantages estimated via Generalized Advantage Estimation (GAE), $\hat{A}_t^{\text{GAE}} = \sum_{l=0}^{\infty}(\gamma\lambda)^l \delta_{t+l}$, $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$, clip parameter $\epsilon = 0.2$, $\lambda = 0.95$, and $K = 4$ epochs of minibatch updates per rollout.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 July 2026 doi:10.20944/preprints202607.2003.v1

A2C.

A2C uses the same actor-critic architecture as PPO but updates online (without clipping or multiple epochs) using the advantage $A(s_t, a_t) \approx \sum_{k=0}^{T-t} \gamma^k r_{t+k} - V(s_t)$ and the policy gradient $\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a \mid s)\, A(s,a)]$.

2.1.3. Hyperparameters and Training Budget

Table 1 summarizes the hyperparameters used for each algorithm. All networks were optimized with Adam. Training was capped at 2000 episodes for DQN, Double DQN, and REINFORCE; 700 episodes for PPO (reduced owing to compute-time constraints); and 8000 episodes for A2C. Episodes were truncated at 1000 steps. All experiments were run on Google Colab, using a GPU (CUDA) where available and CPU otherwise.

Table 1. Hyperparameters used for each algorithm in the LunarLander-v2 study.

Hyperparameter	DQN / DDQN	REINFORCE	PPO	A2C
Learning rate	5×10^{-4}	1×10^{-2}	3×10^{-4}	7×10^{-4}
Discount factor γ	0.99	0.99	0.99	0.99
Batch size	64	–	–	–
Replay buffer size	10^5	–	–	–
Target soft-update τ	10^{-3}	–	–	–
Update frequency	4	–	–	–
ϵ (start/end/decay)	1.0 / 0.01 / 0.995	–	–	–
K epochs	–	–	4	–
Clip ϵ_{clip}	–	–	0.2	–
GAE λ	–	–	0.95	–
Optimizer	Adam	Adam	Adam	Adam

2.2. Study 2: Snow Drone Rescue – A Custom Environment

2.2.1. Motivation

Standard benchmarks such as LunarLander embed carefully engineered, dense reward signals that can mask the difficulty of designing reward functions for realistic tasks. To probe this gap, we designed *Snow Drone Rescue*, a custom, `pygame`-based, Gymnasium-compatible environment in which an autonomous drone must locate and collect stranded targets in a snow-covered mountainous terrain while managing a depleting battery under stochastic environmental disturbance (falling snow).

2.2.2. Environment Design

The environment operates in a continuous two-dimensional arena containing a mobile drone agent, multiple randomly placed rescue items, static obstacles (trees and rocks), and stochastic snowfall. Episodes terminate on battery depletion, collection of all rescue items, or a step-limit timeout.

Partial observability. The agent receives a six-dimensional observation

$$s_t = (x_d, y_d, x_c, y_c, b_t, d_t) \in \mathbb{R}^6,$$

comprising the drone's normalized position (x_d, y_d), the position of the nearest rescue item (x_c, y_c), the normalized remaining battery b_t, and the normalized distance to the nearest item d_t. Information about non-nearest items and obstacles is withheld, enforcing partial observability and discouraging trivial full-map path planning.

Action space. The action space is discrete, $\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}\}$, each moving the drone by a fixed step.

Transition dynamics. Transitions $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ are stochastic through randomized initial item/obstacle placement, altitude-dependent battery drain, obstacle-proximity penalties, and visual (but non-informational) environmental noise from snowfall.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 July 2026 doi:10.20944/preprints202607.2003.v1

2.2.3. Reward Design

The total reward at each step combines dense shaping and sparse event terms:

$$R_t = R_{\text{step}} + R_{\text{progress}} + R_{\text{item}} + R_{\text{collision}} + R_{\text{terminal}}.$$

A constant step penalty $R_{\text{step}} = -0.01$ discourages wandering. A progress-shaping term rewards decreasing distance to the nearest item,

$$R_{\text{progress}} = \begin{cases} +0.5 & d_{t+1} < d_t \\ -0.2 & \text{otherwise,} \end{cases}$$

and sparse terms award $+50$ for item collection, -10 for obstacle collision, $+100$ for full mission completion, and -20 for battery depletion. This combination follows a standard shaping compromise: dense signals accelerate early learning, while sparse terminal rewards dominate the asymptotic policy so as to avoid distorting the task's true objective.

2.2.4. MDP Specification, Algorithm Choice, and Architecture

The task is modeled as an MDP $M = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ with $\gamma = 0.99$. DQN was selected for this environment because of its discrete action space, moderate observation dimensionality, and the sample-efficiency and stability benefits of experience replay and target networks – properties that Study 1 (Section 3) indicates are particularly advantageous under sparse and shaped reward regimes. The Q-network is a feedforward network, $Q_\theta : \mathbb{R}^6 \to \mathbb{R}^4$, with two 128-unit ReLU hidden layers. Training used a replay buffer of 10,000 transitions, batch size 64, learning rate 10^{-3}, and ϵ-greedy exploration with decay, following Algorithm 1.

Algorithm 1 Deep Q-Learning with Experience Replay for Snow Drone Rescue

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value network Q with random weights θ
Initialize target network $\hat{Q}$ with weights $\theta^- = \theta$
for episode $= 1$ to M **do**
  Initialize state s_1 from `SnowDroneEnv.reset()`
  for $t = 1$ to T **do**
    Select $a_t = \arg\max_a Q(s_t, a; \theta)$ with probability $1 - \epsilon$, else a random action
    Execute a_t; observe reward r_t and next state s_{t+1}
    Store transition $(s_t, a_t, r_t, s_{t+1}, \text{done})$ in $\mathcal{D}$
    Sample a random minibatch $(s_j, a_j, r_j, s_{j+1}, d_j)$ from $\mathcal{D}$
    Set $y_j = r_j$ if d_j, else $y_j = r_j + \gamma \max_{a'} \hat{Q}(s_{j+1}, a'; \theta^-)$
    Perform a gradient-descent step on $(y_j - Q(s_j, a_j; \theta))^2$
    Every C steps, set $\theta^- \leftarrow \theta$
  end for
end for

3. Results

3.1. LunarLander-v2 Comparative Benchmark

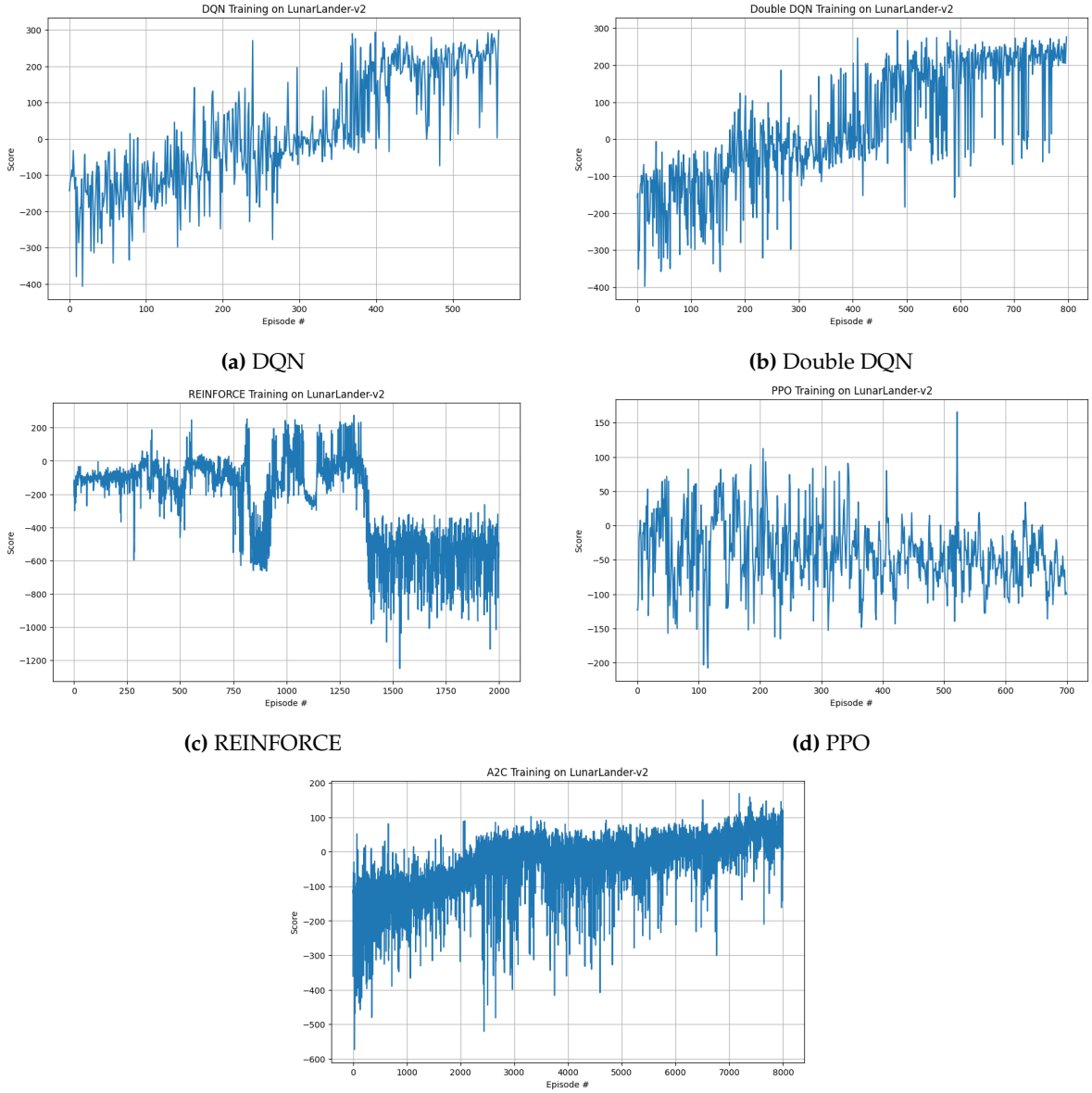
3.1.1. Learning Curves

Figure 1 shows the per-episode score trajectories for all five algorithms. DQN (Figure 1a) exhibits a noisy but clearly monotonic upward trend, solving the environment (average score > 200 over 100 consecutive episodes) in 461 episodes. Double DQN (Figure 1b) follows a similar trajectory with slightly slower initial learning, solving the task in 798 episodes and reaching a comparable final score. REINFORCE (Figure 1c) is markedly unstable: after an initial, noisy plateau near 0, performance collapses after roughly episode 1350 and does not recover, ending at an average score of -565.68.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 July 2026 doi:10.20944/preprints202607.2003.v1

PPO (Figure 1d) improves gradually from strongly negative initial scores but plateaus around −50 to 0 without solving the task within its 700-episode budget. A2C (Figure 1e) shows slow, noisy improvement over a much longer 8000-episode budget, reaching a maximum average score of only 58.



(a) DQN **(b)** Double DQN

(c) REINFORCE **(d)** PPO

(e) A2C

Figure 1. Per-episode learning curves for the five DRL algorithms on LunarLander-v2. DQN and Double DQN converge to stable, high-scoring policies; REINFORCE, PPO, and A2C exhibit substantially higher variance and fail to solve the environment within their respective training budgets.

Figure 2 overlays all five learning curves on a common episode axis, visually emphasizing the disparity in both convergence speed and final variance between the value-based and policy-gradient families.

3.1.2. Performance Summary

Table 2 summarizes the outcome for each algorithm.

DQN and Double DQN were the only algorithms to solve LunarLander-v2 within the allotted budgets, and both display high sample efficiency, which we attribute to their use of a replay buffer (allowing repeated reuse of past transitions) and a target network (which stabilizes the regression target). REINFORCE's on-policy, single-sample Monte-Carlo gradient estimator produces the highest-

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 July 2026 doi:10.20944/preprints202607.2003.v1

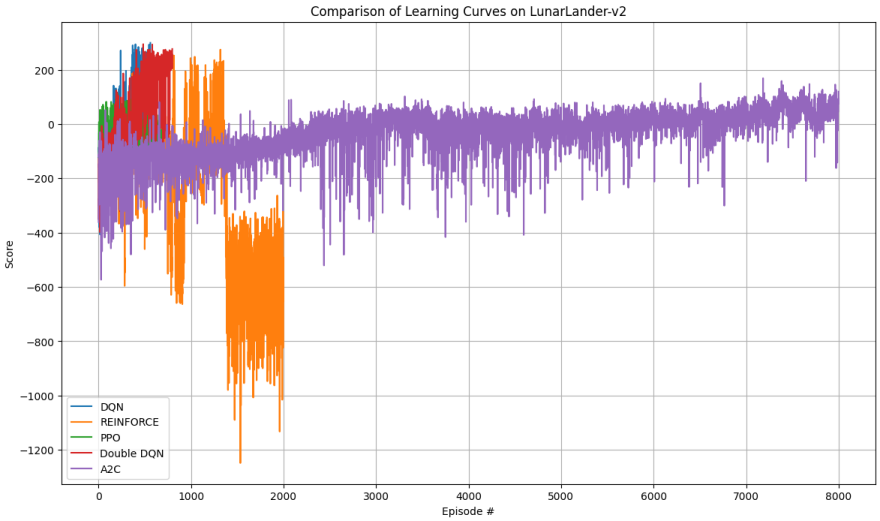


Figure 2. Combined learning curves for DQN, Double DQN, REINFORCE, PPO, and A2C on LunarLander-v2, plotted on a common episode axis.

Table 2. Performance comparison across the five algorithms on LunarLander-v2.

Algorithm	Solved?	Episodes to Solve	Final Avg. Score	Sample Efficiency
DQN	Yes	461	200.87	High
Double DQN	Yes	798	202.32	High
REINFORCE	No	–	−565.68	Very Low
PPO	No	–	−57.67	Low
A2C	No	–	58.00	Very Low

variance updates and, lacking a baseline, is the least sample-efficient of the five methods. PPO and A2C, both of which use a learned critic to reduce variance, outperform REINFORCE but remain well short of solving the task in the given budgets, indicating that variance reduction alone was not sufficient to compensate for the on-policy sample-efficiency disadvantage at these training budgets.

3.1.3. Stability and Convergence

DQN converges with progressively reduced variance as training proceeds; Double DQN follows a similar but slightly smoother trajectory, consistent with its intended reduction of value-overestimation bias. REINFORCE is highly unstable, showing a late-training collapse rather than a monotonic trend. PPO is comparatively more stable than REINFORCE but still exhibits substantial episode-to-episode variability. A2C is relatively stable in trend but converges to a clearly suboptimal policy given its 8000-episode budget.

3.2. Snow Drone Rescue Case Study

Training of the DQN agent in the Snow Drone Rescue environment exhibited three qualitatively distinct phases: (i) an initial phase of largely random exploration accompanied by strongly negative episode returns; (ii) a phase of rapid improvement driven by the dense progress-shaping reward; and (iii) a stabilization phase in which exploration is reduced and the policy converges. Early-training failure modes included battery exhaustion from vertical oscillation, greedy pursuit of the nearest item leading to obstacle collisions, and overfitting to a nearest-item heuristic that ignored longer-horizon item ordering; these observations directly informed subsequent tuning of the reward and exploration-decay schedules.

Qualitative rollouts of the trained policy show energy-aware navigation, with the agent favoring lower-altitude flight paths that reduce battery drain to approximately 0.1 units per step, compared with 0.25 units per step at higher altitudes – a distinction the agent was never explicitly told to make,

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 July 2026 doi:10.20944/preprints202607.2003.v1

but which emerges from the interaction between the battery-linked transition dynamics and the shaped reward. The trained agent also exhibits implicit obstacle avoidance without any explicit map representation, and a tendency toward strategic ordering of rescue targets by proximity, consistent with the intended purpose of the progress-shaping term.

4. Discussion

4.1. Why Value-Based Methods Outperformed Policy-Gradient Methods Here

Three properties of LunarLander-v2 favor value-based learning in our implementation. First, the discrete, low-cardinality action space is naturally suited to Q-learning, which need only compare four scalar action values rather than optimize a full distribution over actions. Second, the environment's reward is comparatively dense and well-shaped (proximity and leg-contact terms provide frequent, informative feedback), which benefits bootstrapped value estimation. Third, off-policy learning with a replay buffer allows DQN and Double DQN to reuse each transition many times, whereas the on-policy algorithms in our comparison discard trajectories after a single (REINFORCE, A2C) or a small, fixed number (PPO, $K = 4$) of gradient updates – a substantial sample-efficiency disadvantage at the training budgets used here.

4.2. Policy-Gradient Failure Modes

REINFORCE's instability follows directly from its reliance on a single-sample Monte Carlo estimator of the policy gradient without a variance-reducing baseline; small unlucky trajectories can produce large, destabilizing gradient steps, and the on-policy constraint prevents the algorithm from amortizing this noise across reused past experience. PPO and A2C mitigate this via a learned critic, but the residual variance, combined with our comparatively short training budgets (700 and 8000 episodes respectively) and default hyperparameters, was insufficient for either to reach the value-based methods' asymptotic performance. Because PPO's clipping mechanism is highly sensitive to ϵ_{clip} and the number of update epochs, and A2C's stability depends on a careful balance between actor and critic learning rates, we expect that a systematic hyperparameter search would narrow, though not necessarily close, the performance gap observed here.

4.3. Reward and Environment Design as a First-Class Concern

The Snow Drone Rescue case study reinforces a complementary point: the choice of algorithm, while consequential, does not act in isolation from the reward function and observation design. A poorly shaped reward in a partially observable setting can produce degenerate behaviours (oscillation, reckless energy expenditure, myopic target selection) regardless of the underlying algorithm's theoretical soundness. This is consistent with observations elsewhere in the DRL literature that reward misspecification and partial observability often dominate algorithmic choice as a source of empirical difficulty in applied, non-benchmark settings.

4.4. Limitations

Several limitations qualify these findings. First, the training budgets for PPO (700 episodes) and A2C (8000 episodes) were not matched in terms of wall-clock time or total environment steps, which complicates direct efficiency comparisons; a fairer comparison would fix a total interaction budget across algorithms. Second, no systematic hyperparameter search was performed for any algorithm; the reported results reflect a single, reasonable hyperparameter configuration per algorithm rather than each algorithm's best achievable performance. Third, results are reported from single training runs per algorithm rather than averaged over multiple random seeds, so the reported learning curves should be interpreted as representative rather than statistically averaged behaviour. Fourth, the Snow Drone Rescue environment, while designed to introduce partial observability and stochasticity, has not been benchmarked against alternative algorithms beyond DQN, so the generality of its qualitative findings across algorithm families remains untested. We view addressing these limitations as natural next steps, outlined below.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 July 2026 doi:10.20944/preprints202607.2003.v1

5. Conclusion and Future Work

Under a matched, from-scratch PyTorch implementation, DQN and Double DQN solved LunarLander-v2 with substantially better sample efficiency and stability than REINFORCE, PPO, or A2C, which is consistent with the theoretical advantages of off-policy replay and target-network stabilization in discrete-action, densely-rewarded environments. A complementary case study on a custom, partially observable Snow Drone Rescue environment shows that a similarly configured DQN agent can learn useful emergent behaviours – energy-aware navigation and implicit obstacle avoidance – from a carefully shaped reward function, but also that reward and environment design choices materially shape what the agent learns, independent of algorithmic sophistication. Together, these results support treating environment and reward engineering as being at least as important as algorithm selection when moving from benchmark tasks to applied autonomous decision-making problems.

Future work will pursue four directions: (i) entropy-regularized and trust-region policy-gradient variants, together with systematic hyperparameter optimization for PPO and A2C, to determine whether the observed gap with value-based methods narrows under a fairer tuning budget; (ii) multi-seed evaluation with statistical significance testing to quantify variance in the reported learning curves; (iii) extension of the comparison to continuous-control benchmarks (e.g., MuJoCo tasks) where policy-gradient methods are typically favored, to test whether the present ranking is specific to discrete-action settings; and (iv) evaluation of alternative exploration strategies (e.g., noisy networks, intrinsic motivation) and recurrent or hierarchical architectures in the Snow Drone Rescue environment, to address its current lack of long-horizon memory and multi-objective planning.

Author Contributions: Conceptualization, S.M. and S.C.; methodology, S.M. and S.C.; software, S.M. and S.C.; validation, S.M. and S.C.; formal analysis, S.M.; investigation, S.M. and S.C.; writing—original draft preparation, S.M.; writing—review and editing, S.M. and S.C.; visualization, S.M. and S.C. All authors have read and agreed to the published version of the manuscript.

Data Availability Statement: The code, trained model checkpoints, and training logs supporting the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments: We thank Tamal Maharaj for guidance on the theoretical foundations of Markov Decision Processes and reward shaping that informed the design of the Snow Drone Rescue environment and the broader experimental work reported here.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533.
2. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016; Vol. 30.
3. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018.
4. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv preprint* **2017**, arXiv:1707.06347.
5. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2016; pp. 1928–1937.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 July 2026 doi:10.20944/preprints202607.2003.v1

6. Towers, M.; Kwiatkowski, A.; Terry, J.; Balis, J.U.; De Cola, G.; Deleu, T.; Goulão, M.; Kallinteris, A.; Krimmel, M.; KG, A.; et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint* **2024**, arXiv:2407.17032.
7. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint* **2015**, arXiv:1506.02438.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.