Reinforcement Learning Proposal

Timo Doorn, van — Miguel Carrasco Guirao — Joost Leliveld — Masanori Nakajima

November 2025

1 Objective and Motivation

This project presents an empirical evaluation of reinforcement learning algorithms in the *LunarLander-v2* environment from Gymnasium. The task models a simplified lunar descent with stochastic dynamics and discrete thruster control, which can be formulated as a Markov Decision Process with continuous states and an action set. Reinforcement learning is appropriate because the agent must optimize sequential decisions under uncertainty and delayed rewards, learning landing strategies directly from interaction without predefined control rules.

Objectives

- Compare the performance and stability of three reinforcement learning algorithms under identical experimental conditions.
- Analyze the sensitivity of these algorithms to hyperparameters such as learning rate, discount factor, and exploration policy.
- Evaluate the consistency and generalization of each method across multiple random seeds.

In the following sections, we present the methodology and results, and determine to what extent the stated objectives are achieved.

2 Environment and MDP Definition

We model the continuous-action environment as a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, T)$ [1].

Each state $s \in \mathcal{S}$ is represented by an 8-dimensional observation vector:

$$s = [x,\, y,\, v_x,\, v_y,\, \theta,\, \dot{\theta},\, c_L,\, c_R],$$

where (x, y) are the horizontal and vertical coordinates of the lander relative to the landing pad center, (v_x, v_y) are the corresponding velocities, θ is the lander's angle, $\dot{\theta}$ its angular velocity, and $c_L, c_R \in \{0, 1\}$ indicate whether the left or right leg is in contact with the ground.

The original environment uses a discrete action space, where you can only do nothing, fire left orientation engine, fire main engine, fire right orientation engine. Using the continuous argument from the environment a continuous two-dimensional action space is defined:

$$\mathcal{A} = [-1, 1]^2,$$

where the first component controls the main engine throttle and the second controls the orientation engine torque. The thrust values are scaled internally to the engine's maximum power. A value of $a = (0, 0)$ corresponds to no thrust, while $a = (1, -1)$ represents full main thrust and full torque to the left.

The transition between states is calculated via the Box2D engine, producing smooth, continuous control dynamics. The next state s' is generated deterministically by the Box2D physics simulator given the current state s and action a, with originally only stochasticity arising from small numerical noise in contact forces and engine jitter. For our use case the environment parameters can be modified to introduce additional uncertainty: the gravitational constant `gravity`, the boolean flag `enable_wind` controlling whether time-varying wind forces are applied, the maximum linear wind strength `wind_power`, and the rotational turbulence strength `turbulence_power`.

The reward structure and termination conditions of the MDP will remain consistent across all policies, using the standard shaped reward provided by the environment to encourage stable and fuel-efficient landings. However, the discount factor γ will be systematically varied to examine its effect on learning stability and robustness under uncertainty.

3 Algorithms and Baselines

We compare several reinforcement learning algorithms on the continuous Lunar Lander environment described above. For learning-based approaches, we first evaluate the performance of Proximal Policy Optimization (PPO), a policy-gradient methods designed for continuous control. This algorithm uses an actor-critic architecture with fully connected multilayer perceptrons for policy and value networks.

Additionally, we adapt two classic value-based algorithms, Deep Q-Networks (DQN) and SARSA. These methods are inherently designed for discrete action spaces. Prior work by [2], for example, compared these two algorithms on the discrete LunarLander-v2 environment, analyzing their performance and robustness under added uncertainty. To adapt these established methods to our continuous problem, we will employ a discretization strategy. This allows us to extend their analysis and directly compare these value-based approaches against the continuous-native PPO algorithm.

Hyperparameters for all learning algorithms are initially chosen from standard configurations used in the literature, and tuned with a small grid search on a validation setting.

4 Experimental Plan

1. First, we compare the three algorithms by running five learning convergence per algorithm for default parameters. This clearly shows the performance and speed of convergence to decide which algorithms under normal circumstances without environmental or action noise is suitable for landing the lunar lander safely within the desired area.
2. We investigate the robustness to environmental uncertainty by introducing environmental noise. We vary a stochasticity parameter σ_e, which introduces wind that affects the movement of the lunar lander in two forms. Wind power dictates the maximum magnitude of linear wind applied to the craft and rotational wind dictates the maximum magnitude of rotational wind applied to the craft. Since, the agents are trained in a noise-free setting ($\sigma = 0$), this tests their ability to cope with degraded control performance.
3. We investigate the robustness to senor noise by introducing observational state noise. Each state s is represented by a 8-dimensional observation vector. six of these observations are continuous and will be randomized by a certain degree of randomness called the stochasticity parameter σ_o. This experiment tests the ability to cope with degraded observational awareness of the lunar lander.
4. Adjusting hyper parameters of the deep reinforcement learning algorithms to measure their affect on performance regarding total reward, success rate and time needed to land.

5 Expected Results and Evaluation Criteria

To assess algorithmic improvement relative to established baselines, we will evaluate the different models based on robustness under uncertainty, sample efficiency, and qualitative behavior.

We will test the trained policies under stochastic perturbations to the environment, including observation noise, randomizing environment parameters, and simulating partial actuator failures. Robustness of a trained model is quantified by the relative performance drop under noisy and nominal conditions. $\Delta R = (\hat{R}_{nominal} - \hat{R}_{noisy})/\hat{R}_{nominal}$.

Besides performance it is also important to compare the training process of the different models. The number of environment interactions required to reach a reward threshold will be used to measure efficiency. And after a quantitative comparison of the performance and the training process we will do qualitative comparison of the policies. Successful policies are expected to exhibit smooth and stable descent trajectories, controlled engine firing, and minimal oscillation near touchdown. Failure modes such as excessive lateral drift, or unstable rotations are unwanted.

6 Potential Difficulties and Mitigation Strategies

We anticipate several practical challenges during experimentation:
Training instability may arise due to the high variance of continuous-control RL methods; this will be mitigated by running multiple seeds and applying standard stabilization techniques such as gradient clipping. Hyperparameter sensitivity is expected across all algorithms, so a small controlled grid search will be performed to ensure fair comparison.

Computational cost is a concern, as each configuration requires repeated training; fixed training budgets, checkpointing, and consistent batch settings will be used to keep experiments tractable. Noise robustness

tests may introduce high variance in performance, which will be addressed by averaging results over multiple evaluation rollouts.
Finally, differences between continuous-action algorithms and discretized methods may affect comparability; identical environment parameters and evaluation procedures will be enforced to maintain fairness.

7 Key Considerations

- The comparison between a continuous-native algorithm (PPO) and value-based methods (DQN, SARSA) relies on our discretization of the continuous action space.
- Since RL algorithms are sensitive to hyperparameter configurations, a fair comparison requires ensuring each algorithm operates near its optimal configuration.
- Training multiple algorithms across various conditions requires substantial computation.

References

[1] Farama Foundation. Lunar lander environment – gymnasium documentation. https://gymnasium.farama.org/environments/box2d/lunar$_l ander/$, 2024. $Accessed : November 2025$.

[2] Soham Gadgil, Yunfeng Xin, and Chengzhe Xu. Solving the lunar lander problem under uncertainty using reinforcement learning. In *2020 IEEE SoutheastCon*, 2020.