TAAP

INSTITUTO UNIVERSITÁRIO DE LISBOA

## Advanced Topics in Deep Learning

MINI-PROJECT 02

# DEEP REINFORCEMENT LEARNING
# DQN AND PPO IN LUNAR LANDER V3

**António Cruz** (140129), **Cátia Brás** (120093), **Ricardo Kayseller** (95813)

# Table of Contents

# 1. Introduction

DQN and PPO are deep reinforcement learning methods that optimize neural network weights through gradient descent, computing how each weight should change to improve performance. This process mimics synaptic plasticity in biological brains, the process where individual connections between neurons are strengthened or weakened during an organism's lifetime based on feedback. As discussed by Sutton & Barto (2018, Chapter 15), this mirrors how dopamine signals in the brain act as a "Temporal Difference" error, allowing an agent to learn which specific actions lead to rewards through trial and error within a single lifespan. This process requires differentiable loss functions, careful learning rate tuning, and mechanisms like experience replay or policy clipping to remain stable.

A genetic algorithm (GA) approaches the same reinforcement learning problem from a fundamentally different angle: phylogenetic rather than ontogenetic learning. Instead of adjusting weights during a single "life" (episode), it treats the neural network's weights as a flat array of numbers, called a "genome," and applies principles borrowed from natural selection to find successful configurations over many generations. While still reinforcement learning, it is not "Deep Learning" in the traditional sense, as the optimization is evolutionary rather than gradient-based. This represents the evolutionary pressure that designs the "innate" circuitry an organism is born with, rather than the learning the organism does after birth.

The process works as follows:

1. *Initialization*: Create a population of 50 random genomes, each representing a complete neural network with random weights.
2. *Evaluation*: Each genome is decoded into a neural network and tested directly in the LunarLander environment. The network receives the same 8 observations (position, velocity, angle, leg contacts) and outputs one of 4 actions, identical to the DQN/PPO networks. Its fitness score is simply the total reward accumulated during the episode.
3. *Selection*: Genomes are ranked by fitness. The top 20% become eligible parents, and the top 3 survive unchanged into the next generation (elitism).
4. *Crossover*: Two parents are selected and their weights are combined to produce a child genome, mixing genetic material from both successful individuals.
5. *Mutation*: Small random perturbations are applied to the child's weights, introducing variation. The mutation rate decays linearly over generations, with large perturbations early for exploration and smaller ones later for refinement.
6. *Repeat*: The new population replaces the old one, and the process repeats for thousands of generations.

The key distinction is what drives the Credit Assignment Problem. In DQN/PPO, the network receives precise mathematical feedback about which direction to adjust each weight for every specific action taken (Temporal Credit Assignment). In the GA, there is no such signal. The algorithm only knows “this set of weights scored 280, that one scored 150” (Global Credit Assignment). It discovers good weights purely through trial, selection, and incremental refinement over generations.

This makes the GA less sample-efficient, requiring roughly 150x more environment interactions than DQN or PPO to reach comparable performance, but also entirely gradient-free. It requires no loss function design, no replay buffer, and no learning rate schedule. The neural network architecture itself is a simple feedforward network with two hidden layers of 10 neurons each—a “minimalist brain” that proves, as Sutton & Barto suggest, that even simple structures can produce complex, goal-directed behavior if the evolutionary search process is sufficiently robust.

The following code imports all required libraries for the genetic algorithm implementation, including numerical computation (NumPy), data handling (pandas), plotting (Matplotlib), and the Gymnasium environment. The custom modules GeneticAlgorithm and NeuralNetwork implement the evolutionary loop and the policy network used by the GA.

# 2. Environment Analysis

## 2.1 State Space

The LunarLander-v3 environment provides an 8-dimensional continuous state vector describing the physical configuration of the lander at each timestep: 1. Horizontal position (x) relative to the landing zone 2. Vertical position (y) above the ground 3. Horizontal velocity (vx) 4. Vertical velocity (vy) 5. Lander angle ($\theta$) 6. Angular velocity ($\omega$) 7. Left leg contact indicator (0 or 1) 8. Right leg contact indicator (0 or 1)

These variables allow the agent to infer its location, orientation, stability, and motion, which are essential for controlling the descent.

## 2.2 Action Space

The action space is discrete with four possible actions:

- 0 - Do nothing
- 1 - Fire left thruster (pushes the lander to the right)
- 2 - Fire main thruster (reduces vertical speed)
- 3 - Fire right thruster (pushes the lander to the left)

These actions allow the agent to control both horizontal movement and vertical descent.

## 2.3 Reward Function

The reward function is dense and designed to encourage smooth and stable landings while penalizing unsafe or inefficient behaviour.

**Positive rewards include:**

- Moving closer to the landing zone
- Reducing horizontal and vertical velocity
- Maintaining a stable angle
- Each leg touching the ground (+10 each)
- Successful landing (+100 to +140)

**Penalties include:**

- High velocities
- Large tilt angles
- Excessive use of the main thruster (fuel cost)
- Crashes (around -100)

A score of 200 or higher typically indicates a successful landing.

## 2.4 Termination Rules

An episode terminates when one of the following occurs:

- **Successful landing** within the designated zone
- **Crash** due to excessive speed or unstable angle
- **Leaving the screen boundaries**
- **Reaching the maximum number of steps** allowed by the environment (1000 timesteps)

These termination conditions apply equally to both PPO and DQN agents.

## 2.5 Global Configuration

```
Session prefix: dqn_ppo
Algorithms: ['dqn', 'ppo']
Seeds: [42, 123, 3407]
Wind enabled: False
Total timesteps per seed: 1,500,000
Evaluation episodes per seed: 20
Chart update frequency: every 10 episodes
Checkpoint frequency: every 100 episodes
Eval callback frequency: every 25,000 timesteps (20 episodes)
Solved threshold: 200
Device: cpu
```

All experiments were conducted under a unified global configuration in order to ensure fairness, consistency, and reproducibility across algorithms and random seeds. Both Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) were trained and evaluated using the same experimental protocol, with identical environment settings, total training budget, and evaluation procedures.

A common session identifier (dqn_ppo) was used to label all trained models and experimental artifacts, ensuring traceability across different runs and seeds. Experiments were performed using three independent random seeds (42, 123, and 3407), allowing the assessment of variability in learning dynamics and final performance.

Each agent was trained for a total of 1.5 million environment interactions per seed. To monitor training progress and convergence behaviour, intermediate evaluations were performed every 25,000 environment steps using a deterministic policy over 20 evaluation episodes. This evaluation protocol provided a consistent basis for comparing learning speed, stability, and asymptotic performance between DQN and PPO. In addition, training checkpoints were saved every 100 episodes to allow inspection of intermediate policies and facilitate qualitative behavioural analysis.

Live training statistics and learning curves were updated every 10 episodes, enabling continuous monitoring of training dynamics and early detection of potential instabilities or plateaus. The standard LunarLander-v3 benchmark criterion was adopted to characterize successful task resolution, with trained agents considered to have reached competent performance when achieving an average return of at least 200 over evaluation episodes.

# 3. Experimental Setup

This section describes the training and evaluation methodology used to compare DQN and PPO in the LunarLander-v3 environment. All experiments were conducted under controlled and reproducible conditions to ensure a fair comparison between the two algorithms.

## 3.1 Total Environment Steps

Both DQN and PPO were trained using a total of 1,500,000 environment steps per seed. Since the project uses 3 seeds, this results in: 3 seeds x 1,500,000 steps = 4,500,000 training steps per algorithm.

Using the same number of interactions ensures that the comparison between DQN and PPO is based on equal sample budgets.

## 3.2 Hyperparameter Configurations

The hyperparameters for each algorithm were selected based on Stable-Baselines3 recommendations and refined through empirical testing. Tables 1 and 2 summarize the final configurations used for training.

Tables 1 and 2 summarize the final configurations used for training.

| Hyperparameter | Value |
|---|---|
| Policy | {MLP_POLICY} |
| Learning rate | linear_schedule(6.3e-4) |
| Buffer size | {dqn_p[‘buffer_size’]:,} |
| Batch size | {dqn_p[‘batch_size’]} |
| Gamma (discount factor) | {dqn_p[‘gamma’]} |
| Learning starts | {dqn_p[‘learning_starts’]:,} |
| Exploration strategy | $\\epsilon$-greedy |
| Exploration fraction | {dqn_p[‘exploration_fraction’]} |
| Final $\\epsilon$ | {dqn_p[‘exploration_final_eps’]} |
| Target update interval | {dqn_p[‘target_update_interval’]} |
| Train frequency | {dqn_p[‘train_freq’]} |
| Gradient steps | {dqn_p[‘gradient_steps’]} |
| Network architecture | {’ x ’.join(str(x) for x in dqn_p[‘policy_kwargs’][‘net_arch’])} MLP |

**Table 1** - DQN Hyperparameters

| Hyperparameter | Value |
|---|---|
| Policy | {MLP_POLICY} |

| Hyperparameter | Value |
|---|---|
| Learning rate | {ppo_p['learning_rate']} |
| n_steps | {ppo_p['n_steps']} |
| Batch size | {ppo_p['batch_size']} |
| n_epochs | {ppo_p['n_epochs']} |
| Gamma (discount factor) | {ppo_p['gamma']} |
| GAE $\\lambda$ | {ppo_p['gae_lambda']} |
| Clip range | {ppo_p['clip_range']} |
| Entropy coefficient | {ppo_p['ent_coef']} |
| Network architecture | {' x '.join(str(x) for x in dqn_p['policy_kwargs']['net_arch'])} MLP |

**Table 2** - PPO Hyperparameters

These hyperparameters were kept constant across all seeds to isolate the effect of stochasticity in training.

## 3.3 Random Seeds

To evaluate robustness and training stability, both algorithms were trained using 3 different random seeds:

- Seed 1: 42
- Seed 2: 123
- Seed 3: 3407

Each seed corresponds to a full independent training run of 1,500,000 steps.

## 3.4 Evaluation Protocol

Each trained model was evaluated over 20 deterministic episodes per seed, for a total of 60 evaluation episodes per algorithm. Deterministic evaluation ensures that the policy is tested without exploration noise, providing a fair assessment of the learned behaviour.

## 3.5 Best-Model Selection

During training, an evaluation callback assessed the model every 25,000 environment steps using 20 deterministic episodes. The best model was selected based on a combined metric: mean reward - standard deviation. This metric favours models that are both high-performing and consistent, rather than models that achieve high mean reward with large variance.

A two-tier selection gate ensures that once any model crosses a mean reward of 200, only solved models can replace the current best.

## 3.6 Reproducibility

All experiments used fixed random seeds for Python, NumPy, PyTorch, and the Gymnasium environment. Deterministic algorithms were enabled in PyTorch to minimize non-determinism from GPU operations.

```
Python: 3.12.3
PyTorch: 2.10.0+cu130
Device: cpu
CUDA: 13.0


Observation space: Box([ -2.5        -2.5       -10.        -10.         -6.28318
55 -10.
  -0.         -0.       ], [ 2.5        2.5       10.        10.         6.283185
5 10.
  1.         1.       ], (8,), float32)
Action space: Discrete(4)
Initial observation: [-0.00170641  1.4186276  -0.17285168  0.3425566   0.00198406
  0.03915349
  0.          0.        ]
```

| Component | Description | Shape | Range (Lower Bound) | Range (Upper Bound) | Type |
|---|---|---|---|---|---|
| Observation (State) | Continuous state vector describing the lander's dynamics and contacts | (8,) | [-2.5, -2.5, -10, -10, -6.28, -10, 0, 0] | [2.5, 2.5, 10, 10, 6.28, 10, 1, 1] | Box(float32) |
| Action | Discrete control commands for the lander engines | (1,) | 0 | 3 | Discrete(4) |

**Table 3** – Observation and Action Spaces of the LunarLander-v3 Environment

The LunarLander-v3 environment provides a continuous state space represented by an 8-dimensional observation vector, capturing the lander's horizontal and vertical position, linear velocities, orientation angle, angular velocity, and binary contact indicators for each landing leg. As shown in Table 3, each state variable is bounded within a predefined range, reflecting physical constraints of the simulated system. These continuous observations require function approximation techniques, making the environment well-suited for deep reinforcement learning approaches.

The action space is discrete and consists of four possible control commands: no action, activation of the left engine, activation of the main engine, and activation of the right engine. This discrete control setting allows for a direct comparison between value-based methods, such as Deep Q-Networks (DQN), and policy-based methods, such as Proximal Policy Optimization (PPO), both of which are applicable to discrete action spaces.

# 4. Training Monitoring, Logging, and Model Selection

To ensure a rigorous analysis of learning dynamics, training stability, and final performance, custom logging and evaluation callbacks were implemented for both DQN and PPO. These callbacks extended the standard Stable-Baselines3 training loop with detailed episode-level tracking, periodic checkpointing, live visualization of learning curves, and the systematic collection of algorithm-specific metrics.

### 4.1.1 Episode-Level Monitoring and Learning Curves

For both algorithms, episode-level rewards and episode lengths were recorded throughout training. This enabled the construction of learning curves depicting the evolution of episodic return and episode duration over time. Rolling averages over recent episodes were used to smooth high-variance signals and facilitate the visual identification of learning phases, plateaus, and potential instabilities. A horizontal reference line corresponding to the solved threshold (average return ≥ 200) was included in the plots to contextualize learning progress with respect to the benchmark performance criterion of the LunarLander-v3 environment.

Live updates of summary statistics were generated at regular intervals, reporting the mean, standard deviation, minimum, and maximum return over recent episodes, as well as the proportion of successful episodes (defined as episodes achieving a return above the solved threshold). This provided immediate feedback on both learning speed and consistency during training.

### 4.1.2 Algorithm-Specific Logging and Stability Metrics

Beyond generic performance indicators, algorithm-specific internal metrics were logged to support a deeper analysis of learning dynamics and theoretical properties.

For the value-based agent, the training loss associated with Q-value regression (Q-loss) was recorded after each rollout. In addition, the exploration rate (ε) was tracked over time to characterize the decay of ε-greedy exploration. To investigate potential overestimation bias, the maximum predicted Q-values for sampled states were periodically logged, providing an approximate proxy for monitoring the magnitude and evolution of Q-value estimates. The total number of gradient updates performed by the DQN agent was also tracked, enabling a fair comparison of optimization effort between algorithms.

For the policy-based agent, multiple loss components were recorded, including policy gradient loss, value function loss, and entropy loss. These metrics allowed the separate inspection of policy improvement, value function fitting, and exploration pressure. In addition, PPO-specific stability indicators were logged, namely the clipping fraction (fraction of updates affected by the clipping mechanism), the approximate Kullback–Leibler (KL) divergence between successive policies, and the explained variance of the value function. Together, these metrics provide insight into the stability and reliability of PPO updates, the extent to which policy updates remain within the trust region imposed by clipping, and the quality of value function approximation over the course of training. As with DQN, the number of gradient updates was tracked to quantify the computational effort invested by the algorithm.

### 4.1.3 Periodic Checkpointing and Qualitative Analysis

To facilitate qualitative evaluation of intermediate policies, training checkpoints were saved periodically at fixed episode intervals (100 episodes). These checkpoints allowed for the visual inspection of agent behaviour at different stages of learning, supporting the qualitative assessment of landing strategies, failure modes, and behavioural improvements over time. Such qualitative analysis complements quantitative metrics and is particularly useful for interpreting learning dynamics in control tasks such as LunarLander, where suboptimal behaviours (e.g., oscillatory hovering or excessive thrust usage) can be readily observed in rollouts.

### 4.1.4 Evaluation Protocol and Best-Model Selection Criterion

In addition to training-time logging, a customized evaluation callback was employed to perform periodic evaluations of the current policy using a deterministic action selection scheme. At each evaluation point, the agent was assessed over a fixed number of episodes, and the mean return, standard deviation of returns, mean episode length, and success rate (percentage of episodes with return ≥ 200) were computed.

To select the best-performing model in a manner that balances performance and consistency, a combined selection metric was defined as:

$$Score = \mu_{return} - \sigma_{return}$$

This criterion favours policies that achieve high average performance while exhibiting low variability across evaluation episodes. Furthermore, a two-tier selection mechanism was implemented: once any evaluated model reached the solved threshold (mean return ≥ 200), only solved models were considered eligible to replace the current best model. Prior to achieving this threshold, the best unsolved model according to the combined score was retained as a fallback. This strategy ensured that final model selection prioritized both competence and stability, in line with the objectives of the project.

## 4.2 Relevance to Evaluation Criteria

The logging and evaluation infrastructure described above directly supports several of the mandatory analysis dimensions required in this project. In particular, it enables: (i) the analysis of learning curves and sample efficiency, (ii) the assessment of training stability across seeds through variance and oscillation patterns, (iii) the empirical investigation of overestimation tendencies in DQN, and (iv) the evaluation of update stability in PPO via clipping fraction and KL divergence. By systematically collecting these metrics, the experimental framework provides a robust basis for the comparative analysis presented in the Results and Discussion sections.

# 5. Training Protocol and Looping Strategy

This section describes the experimental execution strategy adopted to ensure a fair, controlled, and reproducible comparison between DQN and PPO. The training pipeline was designed to systematically iterate over algorithms and random seeds while keeping all other experimental factors fixed. This structured protocol establishes the methodological foundation for the analysis of results presented in the following section, where performance is examined separately for each random seed and subsequently aggregated across seeds.

## 5.1 Looping Strategy Across Algorithms and Seeds

Training was performed by looping over the two selected algorithms (DQN and PPO) and a fixed list of random seeds (42, 123, and 3407). During preliminary experimentation, an additional seed (999) was also tested; however, this configuration failed to consistently reach the solved criterion of an average return ≥ 200 over 100 consecutive episodes within the allocated training budget. Consequently, this seed was excluded from the final comparative analysis and replaced by seed 3407. For each (algorithm, seed) pair, a fresh environment and model instance were created, ensuring independence across runs and preventing information leakage between experiments. This looping strategy enabled the assessment of performance variability and training stability across different random initializations, which is a critical requirement for robust empirical evaluation in reinforcement learning.

Each run was trained for an identical budget of 1.5 million environment interactions. By keeping the total number of environment steps constant across algorithms and seeds, observed differences in learning speed, convergence behaviour, and final performance can be attributed to algorithmic characteristics rather than unequal training budgets.

## 5.2 Controlled Experimental Conditions

All experiments were conducted in the LunarLander-v3 environment with wind perturbations disabled. This ensured that both algorithms were evaluated under identical and controlled environment dynamics, allowing the analysis to focus on intrinsic differences between value-based and policy-based learning mechanisms. A consistent MLP-based policy architecture was used across algorithms, further isolating the effect of the learning algorithm from architectural confounders.

## 5.3 Training-Time Monitoring and Evaluation Protocol

Throughout training, learning curves and summary statistics were continuously monitored using custom logging callbacks. Live statistics were updated at fixed episode intervals, and training checkpoints were periodically saved to support qualitative inspection of intermediate policies. In parallel, periodic evaluations using deterministic policies were conducted at fixed timesteps, providing an objective measure of learning progress and convergence.

This monitoring and evaluation protocol ensures that learning dynamics can be analysed not only in terms of final performance, but also in terms of temporal evolution and stability across different random seeds.

# 6. DQN Results

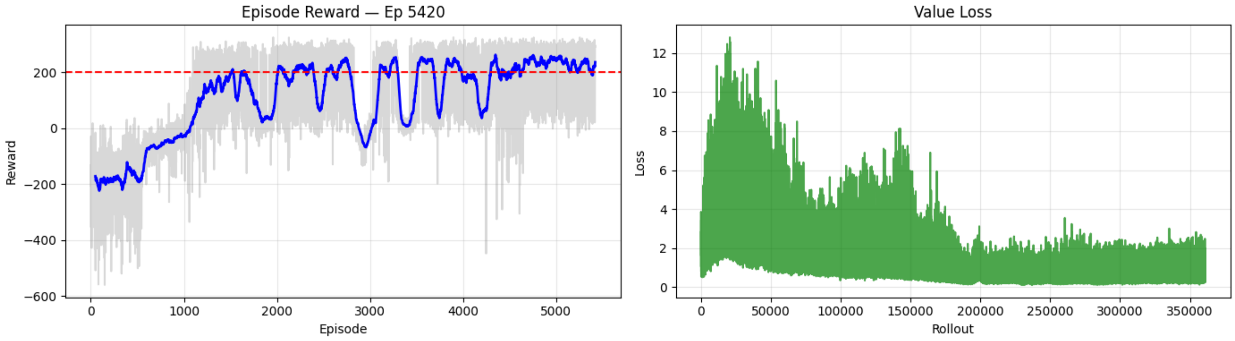


**Figure 1** - DQN training performance (Seed 42)

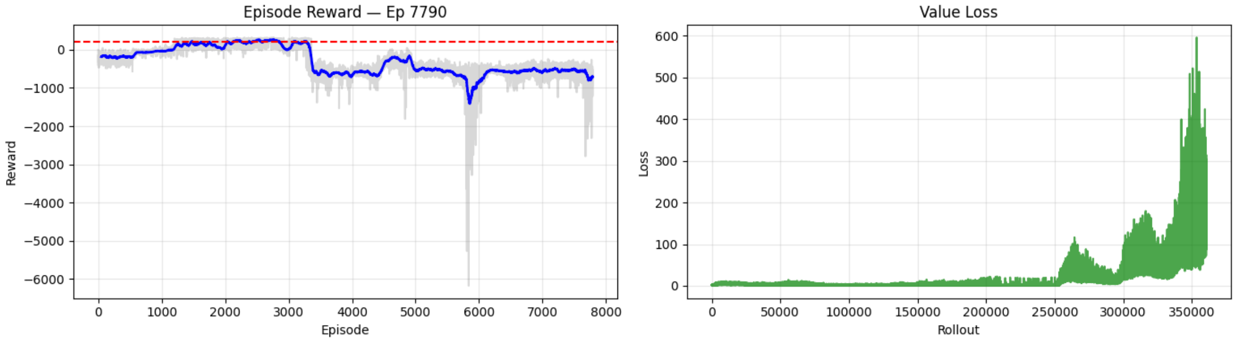


**Figure 2** - DQN training performance (Seed 123)

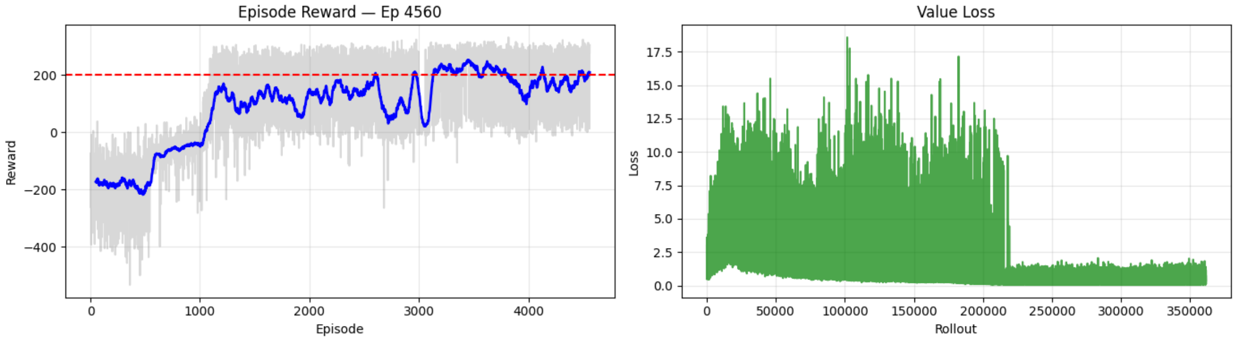


**Figure 3** - DQN training performance (Seed 3407)

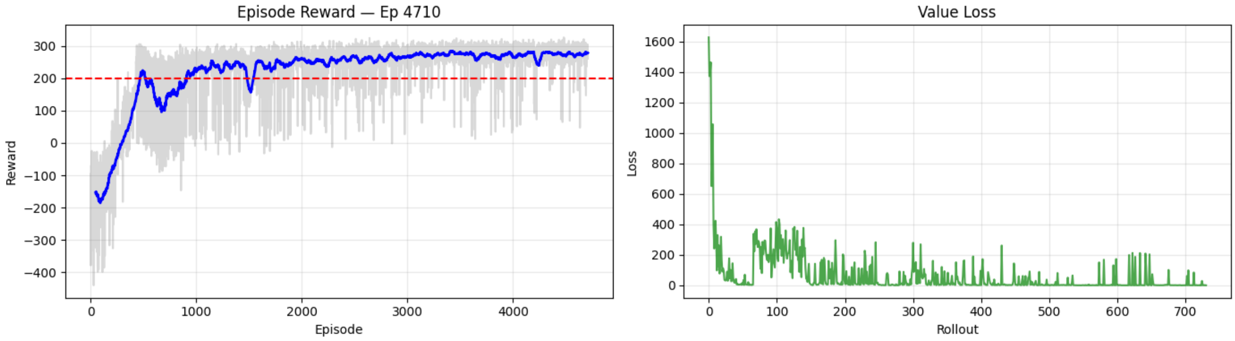


**Figure 4** - PPO training performance (Seed 42)

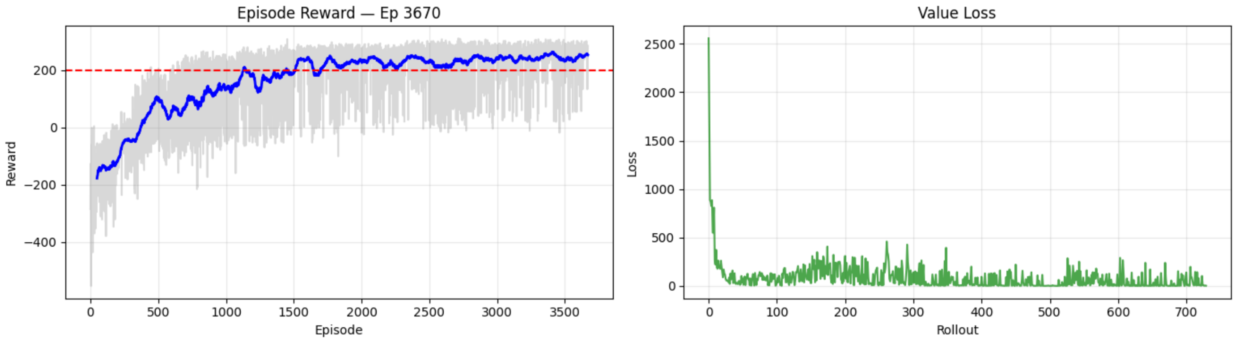


**Figure 5** - PPO training performance (Seed 123)

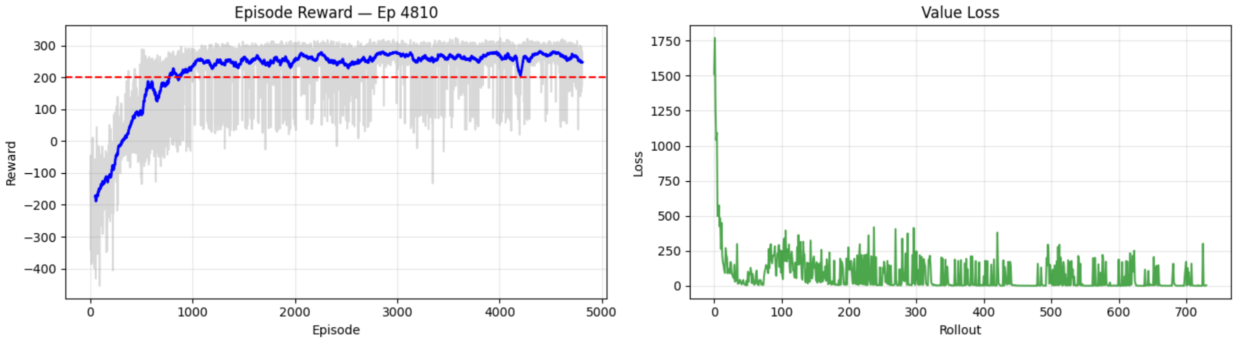


**Figure 6** - PPO training performance (Seed 3407)

| Algorithm | Seed | Mean Reward | Std Reward | Success | Score (Mean – Std) | Best Timestep |
|---|---|---|---|---|---|---|
| DQN | 42 | 283.74 | 17.39 | 100% | 266.35 | 1,425,000 |
| DQN | 123 | 275.61 | 15.80 | 100% | 259.82 | 875,000 |
| DQN | 3407 | 261.53 | 17.66 | 100% | 243.87 | 1,300,000 |
| PPO | 42 | 289.50 | 21.48 | 100% | 268.02 | 1,500,000 |
| PPO | 123 | 267.61 | 18.14 | 100% | 249.47 | 1,300,000 |
| PPO | 3407 | 280.95 | 17.01 | 100% | 263.95 | 1,375,000 |

**Table 4** – Best Model Performance Across Algorithms and Random Seeds

This section reports and analyses the empirical performance of the Deep Q-Network (DQN) across three independent random seeds (42, 123, and 3407). Results are presented per seed to highlight variability in learning dynamics and stability, followed by a cross-seed comparison.

## 6.1 Seed 42

The training run with seed 42 exhibits a clear and successful learning trajectory. Starting from strongly negative episodic returns, the agent progressively improves its landing behaviour and surpasses the solved threshold (average return ≥ 200) after approximately the first quarter of training. The rolling average of episodic reward remains predominantly above the threshold for the remainder of training, despite several pronounced performance drops. These temporary collapses are characteristic of value-based methods and reflect instability in Q-value estimation, likely associated with non-stationary targets and overestimation bias.

The value loss shows high variance and large spikes during early training, followed by a gradual decrease and partial stabilization as the value function becomes better calibrated. Residual oscillations persist even after convergence in performance, indicating that stable loss dynamics are not strictly required for competent policy behaviour in DQN. The best-performing checkpoint for this seed was obtained at approximately 1.425 million environment steps, achieving a mean return of 283.74 ± 17.39 over deterministic evaluation episodes, with a success rate of 100%.

## 6.2 Seed 123

The training dynamics for seed 123 reveal a markedly different and less stable trajectory. While the agent initially approaches and briefly exceeds the solved threshold, learning subsequently degrades, with episodic returns collapsing into strongly negative values for extended periods of training. This catastrophic performance degradation is accompanied by a dramatic explosion in value loss during later stages of training, suggesting severe instability in Q-value updates and divergence in value function approximation.

Despite these instabilities during training, a high-performing checkpoint was identified earlier in the learning process, at approximately 875,000 environment steps. This checkpoint achieved a mean return of 275.61 ± 15.80 with a success rate of 100%. These results highlight a critical property of DQN: strong final performance can be achieved transiently, but training instability may cause subsequent performance collapse if training is continued without early stopping or stability-enhancing mechanisms. This behaviour underscores the importance of checkpointing and periodic evaluation when training value-based deep reinforcement learning agents.

## 6.3 Seed 3407

For seed 3407, the DQN agent demonstrates a comparatively smoother learning progression than seed 123, with episodic returns gradually increasing and stabilizing around the solved threshold. Although fluctuations are still present, the overall trajectory is less volatile than that observed for seed 123. The value loss exhibits an initial high-variance regime followed by a sharp reduction and stabilization, indicating improved convergence of the value function in later training stages.

The best-performing checkpoint for this seed was obtained at approximately 1.300 million environment steps, achieving a mean return of 261.53 ± 17.66 and a success rate of 100%. While the asymptotic performance is slightly lower than that obtained for seeds 42 and 123, the training dynamics appear more stable than the catastrophic collapse observed in seed 123, illustrating the sensitivity of DQN to random initialization and replay buffer dynamics.

## 6.4 Cross-Seed Analysis and Stability of DQN

Across all three seeds, DQN is capable of achieving and surpassing the solved criterion for the LunarLander-v3 environment, demonstrating that value-based deep reinforcement learning can learn competent landing policies under favourable conditions. However, the learning dynamics exhibit substantial variability across seeds. In particular, the instability observed in seed 123, characterized by late-stage performance collapse and exploding value loss, highlights the susceptibility of DQN to divergence and non-stationary training targets.

These results emphasize that DQN performance is highly sensitive to random initialization and stochasticity in experience replay. While high final performance can be achieved, it may not be sustained throughout training, making early stopping and careful model selection crucial. The observed oscillations and occasional collapses are consistent with theoretical concerns regarding overestimation bias and unstable bootstrapping in value-based methods. This motivates the comparison with PPO in the following section, where policy-based learning and clipped updates are expected to yield more stable training dynamics.

# 7. PPO Results

This section reports and analyses the empirical performance of Proximal Policy Optimization (PPO) across independent random seeds. Results are presented per seed to highlight learning dynamics, stability, and final performance, followed by a cross-seed discussion.

## 7.1 Seed 42

The PPO agent trained with seed 42 exhibits a rapid and stable learning trajectory. The episodic return curve shows a smooth improvement during the early phase of training, surpassing the solved threshold (average return ≥ 200) within the first few hundred episodes. Unlike the DQN runs, the learning trajectory of PPO is markedly more stable, with fewer severe performance drops and a smoother convergence towards high average returns.

The rolling average of episodic reward remains consistently above the solved threshold for the majority of training, indicating robust convergence to a competent landing policy. The loss dynamics further corroborate this stability: the value loss exhibits a sharp decrease during early training followed by relatively low and bounded fluctuations. The absence of late-stage loss explosions or catastrophic performance collapses suggests that the clipped policy updates and on-policy learning regime of PPO effectively mitigate the instability issues commonly observed in value-based methods.

Quantitatively, the best-performing PPO checkpoint for seed 42 was obtained at the end of training (1.5 million environment steps), achieving a mean return of 289.50 ± 21.48 over deterministic evaluation episodes. This corresponds to a success rate of 100% and a combined score (mean − standard deviation) of 268.02. Compared to the DQN results for the same seed, PPO achieves slightly higher average performance with a smoother and more reliable learning trajectory, albeit with a somewhat higher variance in final returns.

## 7.2 Seed 123

The PPO agent trained with seed 123 demonstrates a stable and monotonic learning progression. The episodic return increases smoothly from negative initial values and crosses the solved threshold after the early training phase. No catastrophic drops in performance are observed, and the learning curve exhibits limited oscillations compared to the DQN counterpart.

The value loss decreases rapidly during the initial optimization phase and remains bounded throughout training, with moderate fluctuations reflecting ongoing policy and value function refinement. The stability of both reward and loss trajectories indicates that PPO’s clipped objective and advantage estimation mechanisms effectively constrain destructive policy updates.

The best-performing checkpoint for this seed was obtained at approximately 1.3 million environment steps, achieving a mean return of 267.61 ± 18.14 with a success rate of 100%.

While the final average performance is slightly lower than that obtained for seed 42, the training dynamics remain smooth and robust.

## 7.3 Seed 3407

For seed 3407, PPO again exhibits rapid convergence and stable learning dynamics. The episodic return surpasses the solved threshold relatively early in training and stabilizes around high return values thereafter. The learning curve displays minor oscillations but no evidence of divergence or late-stage performance degradation.

The value loss shows an initial transient peak followed by a sharp reduction and sustained low-amplitude fluctuations, consistent with effective value function learning and stable policy optimization. This behaviour contrasts with the exploding losses observed in some DQN runs and highlights the robustness of PPO across different random initializations.

The best-performing checkpoint for this seed was obtained at approximately 1.375 million environment steps, achieving a mean return of 280.95 ± 17.01 with a success rate of 100%. This performance is comparable to the best DQN runs while being achieved under significantly more stable training dynamics.

## 7.4 Cross-Seed Analysis and Stability of PPO

Across all three random seeds, PPO consistently reaches and surpasses the solved threshold for the LunarLander-v3 environment, achieving 100% success rates in final deterministic evaluations. Learning curves are smoother and more monotonic than those observed for DQN, and no catastrophic training collapses are observed across seeds.

While final average returns are broadly comparable between PPO and DQN, PPO demonstrates superior robustness to random initialization and training stochasticity. The reduced variance in learning dynamics, bounded loss signals, and absence of late-stage divergence provide strong empirical evidence that PPO offers a more stable and reliable training process for this control task.

# 8. DQN vs PPO Comparative Analysis

This section provides a direct comparative analysis between Deep Q-Network (DQN) and Proximal Policy Optimization (PPO), integrating quantitative results across seeds with qualitative observations of learning dynamics. The comparison focuses on final performance, sample efficiency, training stability, and robustness to random initialization, and relates the empirical findings to the theoretical properties of value-based and policy-based reinforcement learning methods.

## 8.1 Final Performance Across Seeds

Both DQN and PPO achieved the solved criterion for the LunarLander-v3 environment across all evaluated seeds, with 100% success rates in deterministic evaluations. In terms of mean episodic return, PPO slightly outperformed DQN for seeds 42 and 3407, while DQN achieved competitive performance for seed 123. However, when accounting for variability via the combined score (mean − standard deviation), PPO consistently achieved higher or comparable stability-adjusted performance, indicating a better trade-off between high returns and consistency.

These results suggest that, while both algorithms are capable of learning competent landing policies, PPO tends to deliver more reliable final performance across different random initializations.

## 8.2 Sample Efficiency and Convergence Behaviour

The timestep at which the best-performing checkpoint was obtained varies substantially for DQN across seeds, ranging from early convergence (875k steps for seed 123) to later-stage convergence (1.3–1.4M steps for seeds 42 and 3407). This heterogeneity indicates that DQN's convergence speed is highly sensitive to stochastic factors such as initial network weights, exploration trajectories, and replay buffer composition.

In contrast, PPO exhibits more homogeneous convergence behaviour across seeds, with best checkpoints consistently emerging in the later stages of training (1.3–1.5M steps). Although PPO does not converge earlier in absolute terms, its learning curves are smoother and more predictable, reducing the need for aggressive early stopping and extensive checkpoint selection. From a practical perspective, PPO's more stable convergence profile simplifies experimental management and hyperparameter tuning.

## 8.3 Training Stability and Learning Dynamics

A key empirical distinction between the two algorithms lies in training stability. DQN displays pronounced oscillations in episodic return and, in some runs, catastrophic late-stage performance collapses accompanied by exploding value loss (notably for seed 123). These phenomena are consistent with theoretical concerns regarding overestimation bias, bootstrapped target propagation, and non-stationary learning targets in value-based methods.

By contrast, PPO demonstrates markedly more stable learning dynamics across all seeds. Learning curves are smoother and more monotonic, loss signals remain bounded, and no catastrophic collapses are observed. The clipped surrogate objective and the use of Generalized Advantage Estimation (GAE) effectively constrain policy updates, mitigating the risk of destructive parameter changes. These empirical observations align with PPO’s design objective of improving stability relative to unconstrained policy gradient methods and off-policy value-based approaches.

## 8.4 Exploration Mechanisms and Their Empirical Impact

The observed differences in learning dynamics are further influenced by the exploration strategies employed by each algorithm. DQN relies on ε-greedy exploration, which injects uniform random actions irrespective of state uncertainty. While effective for ensuring coverage of the action space, this strategy can lead to inefficient or destabilizing exploration in later training phases, contributing to abrupt performance drops.

PPO employs entropy-regularized exploration, encouraging stochasticity in the policy distribution in a state-dependent manner. This results in more structured and adaptive exploration, which empirically manifests as smoother learning curves and reduced sensitivity to random initialization. The sustained entropy regularization helps prevent premature policy collapse and supports continuous, controlled exploration throughout training.

## 8.5 Practical Trade-offs and Summary

Both DQN and PPO are capable of achieving high final performance on the LunarLander-v3 task. However, PPO demonstrates superior robustness and training stability across random seeds, with smoother convergence and fewer pathological training behaviours. DQN can achieve competitive or even superior peak performance under favourable conditions, but its sensitivity to initialization and non-stationary learning dynamics necessitates careful monitoring, checkpointing, and early stopping.

From a practical standpoint, PPO emerges as the more reliable and reproducible choice for this control task, particularly in settings where training stability and robustness are prioritized. DQN remains a viable alternative, especially when simplicity or off-policy data reuse is advantageous, but its empirical behaviour underscores the importance of stability-enhancing mechanisms and rigorous experimental control.

# 9. Training Time and Practical Efficiency

Table 5 summarizes the wall-clock training times measured for each algorithm and seed under an identical training budget of 1.5 million environment steps per run, executed on CPU. A clear and consistent pattern emerges: PPO required substantially less training time than DQN across all seeds.

| Algorithm | Seed | Time (s) | Time (min) |
|---|---|---|---|
| DQN | 42 | 2856 | 47.6 |
| DQN | 123 | 2818 | 47.0 |
| DQN | 3407 | 2839 | 47.3 |
| PPO | 42 | 1211 | 20.2 |
| PPO | 123 | 1380 | 23.0 |
| PPO | 3407 | 1248 | 20.8 |
| DQN | Mean | 2838 | 47.3 |
| PPO | Mean | 1280 | 21.3 |

**Table 5** – Training Time Summary (1,500,000 Timesteps per Seed, CPU)

On average, DQN took approximately 47.3 minutes per seed, whereas PPO required approximately 21.3 minutes per seed. This corresponds to PPO being roughly 2.2× faster in wall-clock time under the same number of environment interactions. Importantly, this speed difference was stable across seeds, suggesting that it primarily reflects algorithmic and implementation-level characteristics rather than stochastic variation in episode trajectories.

From a practical standpoint, these results indicate that PPO provides a more computationally efficient pipeline for iterative experimentation and hyperparameter tuning in this setting. Given that reinforcement learning typically requires multiple runs across seeds and multiple hyperparameter configurations, the reduction in wall-clock time can translate into a significantly larger experimental capacity within the same time budget.

This difference is consistent with the underlying training mechanics of both methods. DQN performs frequent off-policy updates using replayed transitions, and the chosen configuration (including multiple gradient steps per training interval) increases the overall optimization workload. PPO, while also performing multiple epochs of optimization per rollout, benefits from efficient on-policy batch processing and a more stable training regime, which in practice resulted in substantially faster end-to-end training times in the present setup.

When considering not only final performance but also computational cost and iteration speed, PPO demonstrated a clear advantage over DQN in this project's CPU-based experimental setting.

## 10. Gradient Updates and Optimization Workload

To complement the comparison based on environment steps (sample budget), we also analysed the number of gradient updates performed by each algorithm, as required by the assignment. Table 6 summarizes (i) the update count recorded from training logs and (ii) an analytical estimate derived from the hyperparameter configuration.

| Algorithm | Actual (from training) | Analytical (computed) | Total Env Steps | Ratio (updates/steps) |
|---|---|---|---|---|
| DQN | 1,449,996 | 1,450,000 | 1,500,000 | 0.967 |
| PPO | 7,320 | 234,240 | 1,500,000 | 0.005 |

**Table 6** – Gradient Updates Summary (per 1,500,000 Environment Steps)

### *DQN: Expected and Observed Updates Are Consistent*

For DQN, the configuration used train_freq = 4 and gradient_steps = 4, which corresponds to performing 4 gradient updates every 4 environment steps once learning begins. Considering the warm-up period (learning_starts = 50,000), the number of effective training steps is 1,500,000 − 50,000 = 1,450,000. Under this setup, the analytical estimate is:

- Training calls ≈ 1,450,000 / 4 = 362,500
- Updates per call = 4
- Total updates ≈ 362,500 × 4 = 1,450,000

The logged value (1,449,996) is virtually identical, confirming that the DQN update accounting is correct and consistent with the intended training schedule. This also indicates that DQN performed a very high optimization workload relative to environment interaction, with an update/step ratio close to 1.0 (≈ 0.967).

### *PPO: Logged Update Count vs Analytical Estimate - Accounting Definition Matters*

For PPO, the analytical estimate assumes that one rollout of n_steps = 2048 is followed by n_epochs = 10 passes over the rollout buffer using mini-batches of size batch_size = 64. This implies:

- Mini-batches per epoch = 2048 / 64 = 32
- Updates per rollout = 10 × 32 = 320
- Number of rollouts ≈ 1,500,000 / 2048 ≈ 732
- Analytical total ≈ 732 × 320 = 234,240 updates

However, the logged “Actual (from training)” value is much smaller (7,320). This discrepancy indicates that the training logger variable used to count updates for PPO does not correspond to “mini-batch gradient steps” in the same way as the analytical calculation. In practice, Stable-Baselines3 may report update counters at a different granularity (e.g., counting policy update iterations or rollout-level update events rather than each mini-batch optimization step), depending on the internal logging definition.

Therefore, while the analytical computation represents the theoretical number of mini-batch optimization steps implied by the PPO configuration, the logged value reflects the update counter as defined internally by the SB3 training loop. As a result, the two numbers are not directly comparable unless both algorithms use the same definition of “gradient update” (e.g., explicitly counting optimizer steps at the mini-batch level).

### *Implications for Fair Comparison*

Even with this accounting nuance, the comparison remains informative in two ways:

1. DQN performs frequent off-policy updates throughout training, leading to a high optimization workload per environment step. This is consistent with its reliance on replay-buffer sampling and repeated bootstrapped regression updates.
2. PPO concentrates optimization into rollout-based training phases, which is reflected in a much lower logged update/step ratio when using SB3’s internal update counter. This aligns with PPO’s on-policy design, where data are collected in batches and then used for a finite number of optimization epochs.

In the next section, these optimization workload differences are interpreted jointly with wall-clock training time and learning stability. Notably, PPO achieved comparable or better final performance with substantially lower wall-clock time, supporting its practical efficiency in this experimental setting.

## 10.1 DQN vs PPO Episode Reward Curves

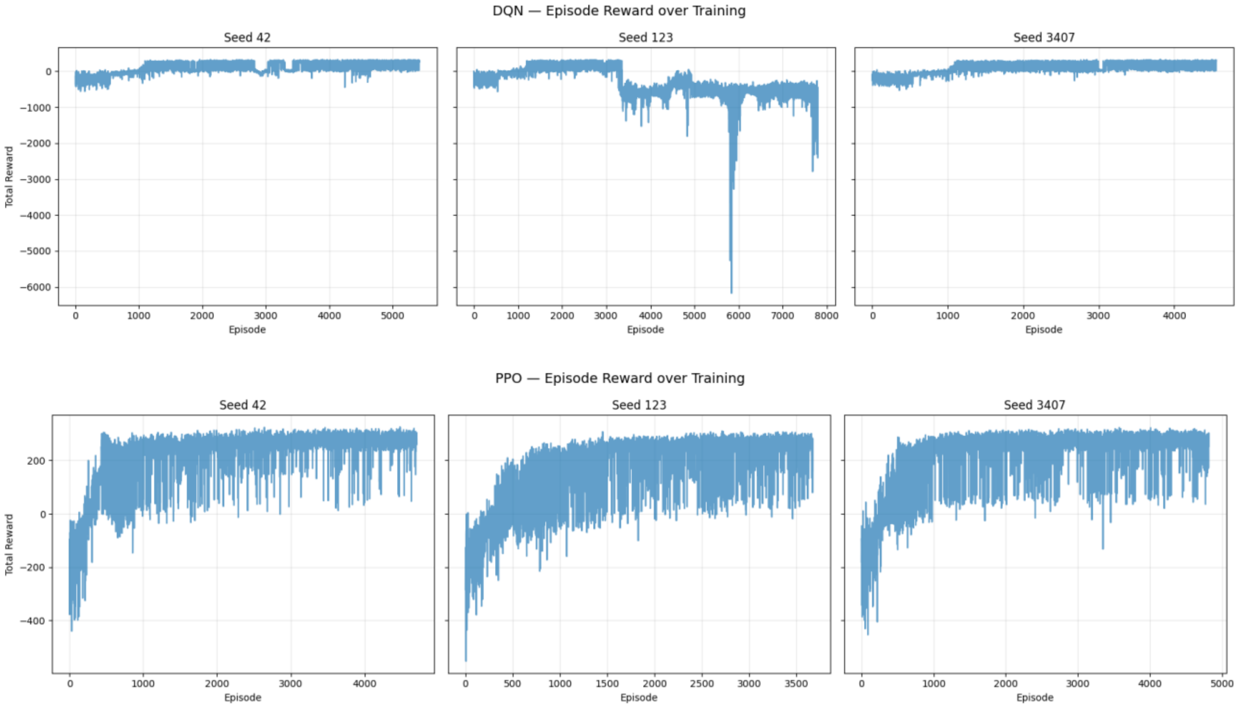


**Figure 7** - Episode reward over training for DQN (top row) and PPO (bottom row)

Figure 7 presents the episodic return trajectories for DQN across seeds 42, 123, and 3407, while Figure Y reports the same analysis for PPO. These plots provide a direct visual comparison of learning dynamics, stability, and sensitivity to random initialization.

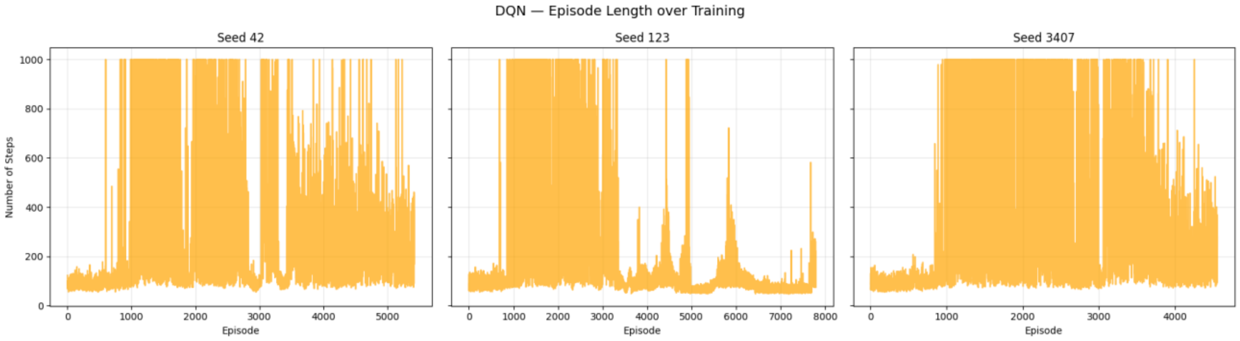
The DQN reward curves show substantial variability across seeds. For seeds 42 and 3407, the agent gradually improves from negative returns to sustained positive performance, stabilizing near or above the solved threshold for large portions of training. However, the curves contain noticeable oscillations and transient drops, consistent with the instability typically associated with bootstrapped Q-learning and replay-buffer dynamics.

The most striking behaviour occurs for seed 123, where training initially improves and briefly reaches competent performance, but later suffers a clear collapse. Returns fall into highly negative regimes, including extreme crash episodes with very large negative rewards. This pattern strongly suggests instability in value estimation (e.g., error amplification through bootstrapped targets and shifting data distributions inside the replay buffer), and it illustrates why checkpointing and periodic evaluation were essential for selecting the best DQN model before collapse.

In contrast, the PPO curves show a more homogeneous pattern across seeds. All seeds exhibit a relatively rapid rise in episodic return early in training, followed by stabilization in a high-performing regime. While episodic variance remains visible (reflecting stochastic transitions and occasional imperfect landings), PPO does not exhibit catastrophic collapses comparable to DQN seed 123. The absence of late-stage divergence aligns with PPO's clipped updates and on-policy training design, which constrain destructive parameter changes and improve optimization stability.

Figure 7 visually reinforce the quantitative findings reported in the best-model table: both algorithms can reach high performance, but PPO displays stronger robustness and stability across seeds, whereas DQN is more sensitive to random initialization and can experience training degradation after initially strong performance. This difference provides strong empirical support for the expected theoretical contrast between value-based learning (susceptible to overestimation and bootstrapping instability) and policy-based actor–critic optimization with constrained updates (PPO).

## 10.2 Episode Length Curves Across Seeds (DQN vs PPO)



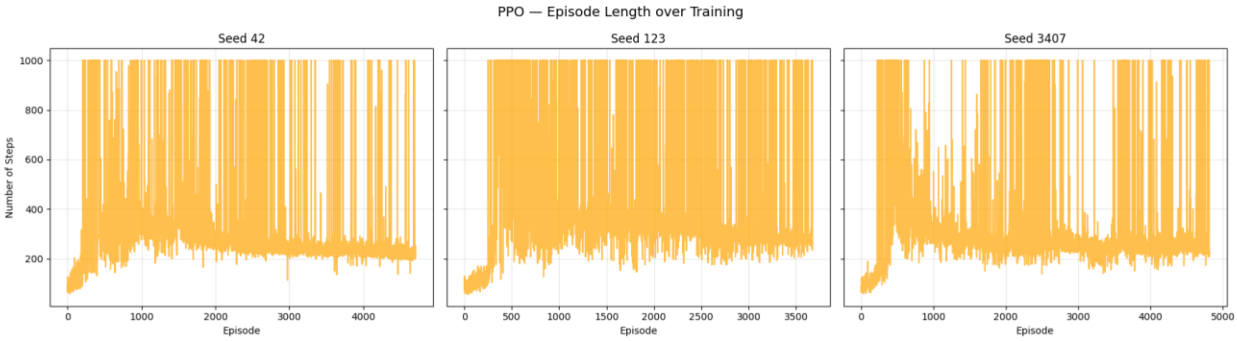


**Figure 8** - Episode length during training for DQN (top) and PPO (bottom) across seeds 42, 123, and 3407.

Figure 8 report the episode length (number of environment steps per episode) throughout training for DQN and PPO, respectively, across seeds 42, 123, and 3407. Episode length provides a complementary behavioural signal to episodic reward: in LunarLander, longer episodes often reflect sustained flight/hovering and repeated corrections (which can be either beneficial—controlled descent—or inefficient—hovering without committing to landing), whereas short episodes frequently correspond to rapid termination, often due to crashes or unstable landings.

For DQN, the episode length dynamics vary substantially across seeds. Seeds 42 and 3407 exhibit extended periods with high episode lengths (often approaching the environment's maximum horizon), indicating that the agent learned to remain airborne and execute longer control sequences. Over training, these long episodes become more structured, consistent with improved stabilization and approach manoeuvres prior to landing. However, the presence of intermittent drops in episode length suggests episodes that terminate early, which is consistent with the reward curve oscillations and occasional failures.

Seed 123 stands out as the least stable run. After an initial phase where long episodes occur frequently (suggesting partial competence), the trajectory becomes erratic with repeated bursts and sharp drops in episode length. This pattern aligns with the reward collapse observed for this seed: shorter episodes occurring later in training are consistent with frequent crashes or failure modes that terminate the episode prematurely. Together, these signals reinforce that DQN can experience training degradation after initially improving, reflecting instability in value estimation and policy quality.

For PPO, episode length curves are more homogeneous across seeds. All runs rapidly transition from shorter episodes (early crashes/instability) to longer episodes as the policy learns to stabilize the lander and manage descent. Episode lengths remain broadly high throughout training, with fluctuations that are consistent with occasional imperfect landings but without the sustained late-stage shortening patterns observed in DQN seed 123. This indicates that PPO not only improves reward but also maintains a stable control regime over time, avoiding collapse into short, failure-dominated episodes.

The episode length plots reinforce the central stability conclusion: PPO achieves more consistent behavioural control across seeds, sustaining long-horizon interaction patterns indicative of stable descent and landing strategies. In contrast, DQN shows stronger seed sensitivity, and the instability observed for seed 123 manifests clearly as late-stage volatility and frequent short episodes, consistent with crash-heavy behaviour. These observations support the broader comparative analysis: PPO's constrained updates and on-policy learning produce more robust training dynamics, whereas DQN remains susceptible to instability and performance collapse in certain seeds.

## 10.3 Value Loss Dynamics Across Seeds (DQN vs PPO)

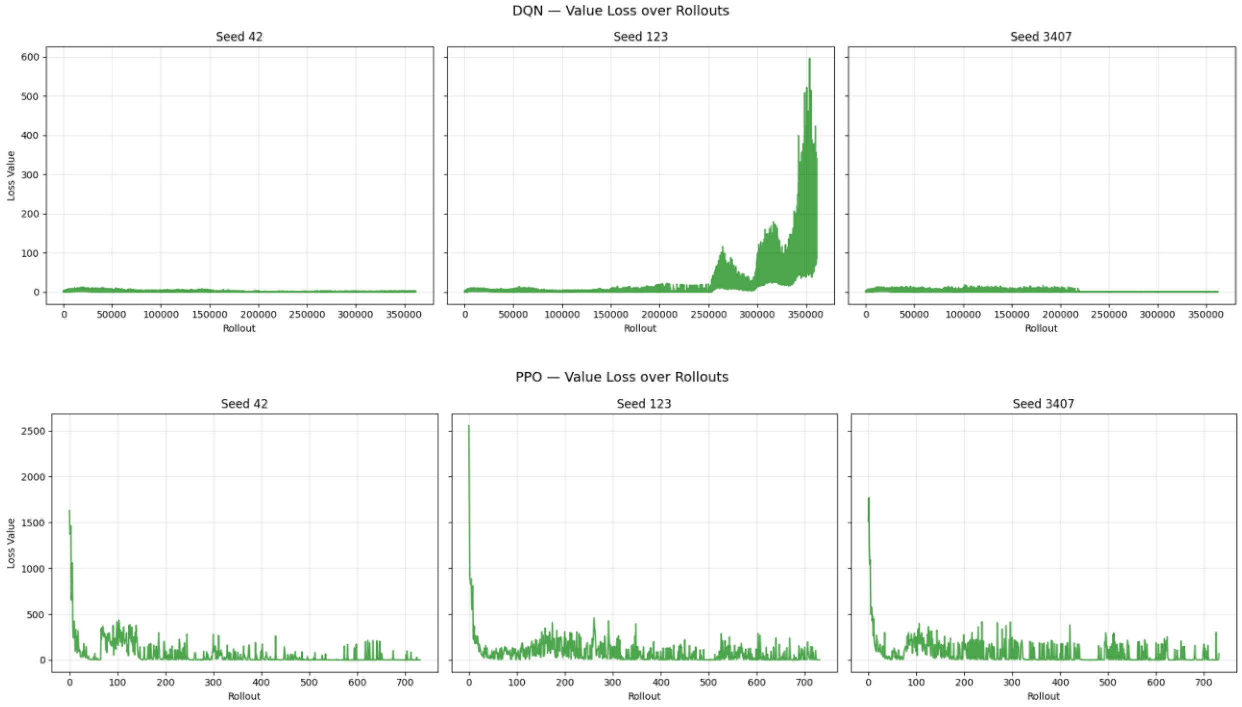


**Figure 9** - Value loss during training for DQN (top) and PPO (bottom) under seeds 42, 123, and 3407

Figure 9 depict the evolution of the value loss over training rollouts for DQN and PPO, respectively, across seeds 42, 123, and 3407. Although value loss is not a direct proxy for policy quality, its dynamics provide important insight into optimization stability, convergence behaviour, and potential divergence phenomena during training.

For DQN, the value loss trajectories vary substantially across seeds. Seeds 42 and 3407 exhibit relatively well-behaved loss dynamics: after an initial phase with elevated variance, the loss gradually decreases and stabilizes at low magnitudes, suggesting improved calibration of Q-value estimates. However, residual oscillations persist, reflecting the inherent noise of bootstrapped value regression under off-policy sampling.

In stark contrast, seed 123 displays a pronounced late-stage explosion in value loss. After a period of moderate stabilization, the loss increases sharply towards the end of training, reaching very large magnitudes. This behaviour is strongly correlated with the catastrophic performance collapse observed in the reward curves and the shortening of episode lengths for this seed. The divergence in value loss indicates instability in Q-value updates, likely driven by error amplification through bootstrapped targets and non-stationary data distributions in the replay buffer. This phenomenon exemplifies the vulnerability of value-based methods to destabilizing feedback loops in deep reinforcement learning.

For PPO, the value loss curves across all seeds share a common pattern: a very high initial peak during early training, followed by a rapid decay as the value function begins to approximate returns more accurately. Subsequently, the loss remains bounded with moderate fluctuations throughout training. Importantly, no late-stage loss explosions are observed for any seed.

The consistency of PPO’s value loss dynamics reflects the stabilizing effect of on-policy data collection and constrained policy updates. While the critic continues to be updated throughout training, the magnitude of the loss remains controlled, suggesting that value function learning does not diverge even as the policy improves. This stability stands in contrast to the divergent behaviour observed for DQN seed 123 and provides further empirical support for PPO’s robustness.

Taken together, the value loss plots reinforce the broader comparative findings: PPO exhibits more stable optimization dynamics across seeds, whereas DQN is more susceptible to instability and late-stage divergence in certain runs. These empirical observations align with theoretical expectations: off-policy bootstrapped value learning in DQN is prone to overestimation bias and error propagation, while PPO’s clipped objective and advantage-based updates help regulate learning and prevent destructive parameter updates.

## 10.4 Exploration Dynamics

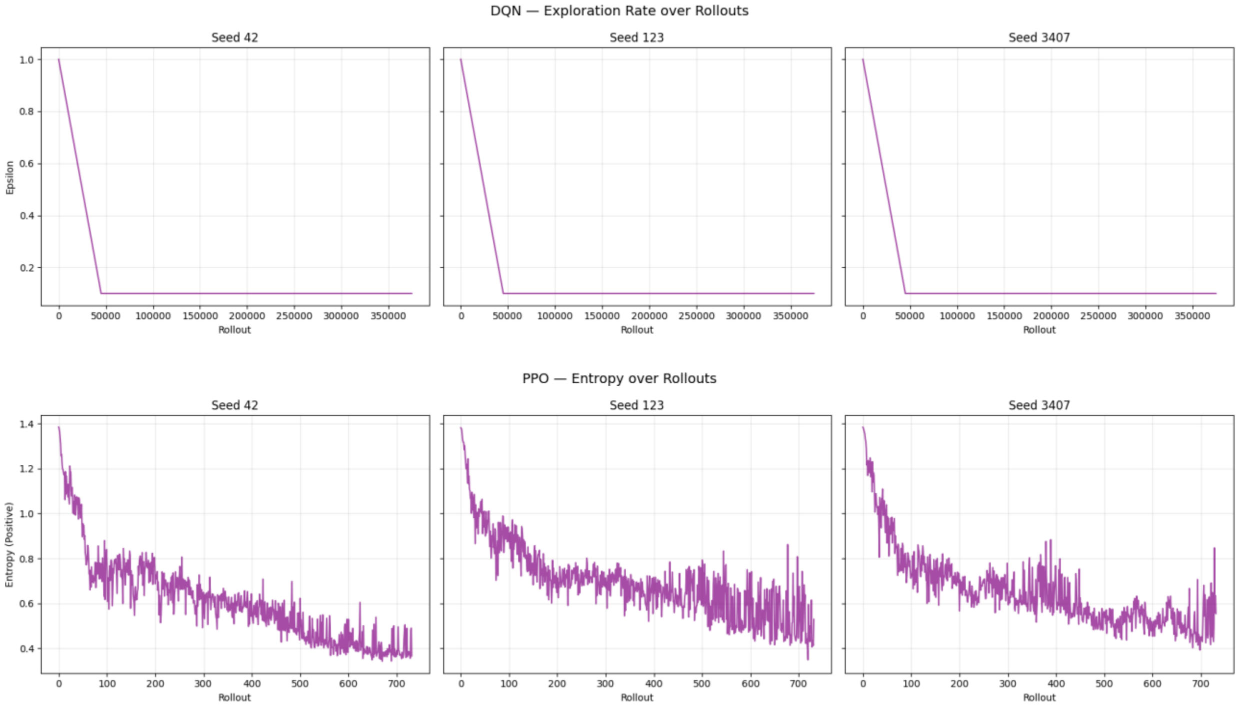


**Figure 10** - Exploration rate (DQN, top) and entropy (PPO, bottom) across seeds 42, 123, and 3407.

Figure 10 illustrate the evolution of exploration-related metrics during training: the ε-greedy exploration rate for DQN and the policy entropy for PPO, across seeds 42, 123, and 3407. These curves provide insight into how each algorithm transitions from exploration to exploitation and how exploration pressure is maintained over time.

For DQN, the exploration rate ε follows a predefined linear schedule, decreasing rapidly from its initial value to the final exploration floor (ε_final = 0.1) within the early phase of training. This behaviour is identical across seeds, as the ε schedule is deterministic and independent of learning progress. After reaching the minimum exploration rate, ε remains constant for the remainder of training, implying that a fixed proportion of actions (10%) continue to be selected uniformly at random.

This rigid exploration schedule implies that the degree of exploration does not adapt to the agent's uncertainty or performance. Consequently, once ε has decayed, the agent's behaviour becomes increasingly exploitative, and any subsequent performance degradation (as observed for DQN seed 123) cannot be counteracted by adaptive increases in exploration. This highlights a limitation of ε-greedy exploration in complex control tasks, where exploration needs may change over the course of training.

For PPO, the policy entropy decreases gradually over training for all seeds, reflecting the transition from highly stochastic policies to more deterministic ones as learning progresses. Unlike ε-greedy, entropy is not externally scheduled but emerges from the interaction between policy optimization and entropy regularization. The smooth decay of entropy indicates that PPO progressively reduces randomness while still maintaining non-zero stochasticity, enabling continued local exploration around promising policies.

Importantly, entropy remains strictly positive throughout training and does not collapse to zero. This sustained exploration pressure helps prevent premature convergence to suboptimal deterministic policies and contributes to PPO's observed training stability. Minor fluctuations in entropy reflect ongoing policy refinement and local exploration, rather than abrupt regime changes.

The contrasting exploration dynamics align with the observed differences in training stability between DQN and PPO. DQN's fixed ε schedule enforces an abrupt transition from exploration to exploitation, which may exacerbate instability once the agent relies heavily on potentially misestimated Q-values. PPO's entropy-driven exploration, by contrast, provides a more gradual and adaptive reduction in stochasticity, supporting stable convergence and reducing the risk of catastrophic policy collapse. These empirical patterns support the theoretical distinction between externally imposed ε-greedy exploration and intrinsically regulated entropy-regularized policy learning.

## 10.5 PPO Policy Gradient Loss Dynamics

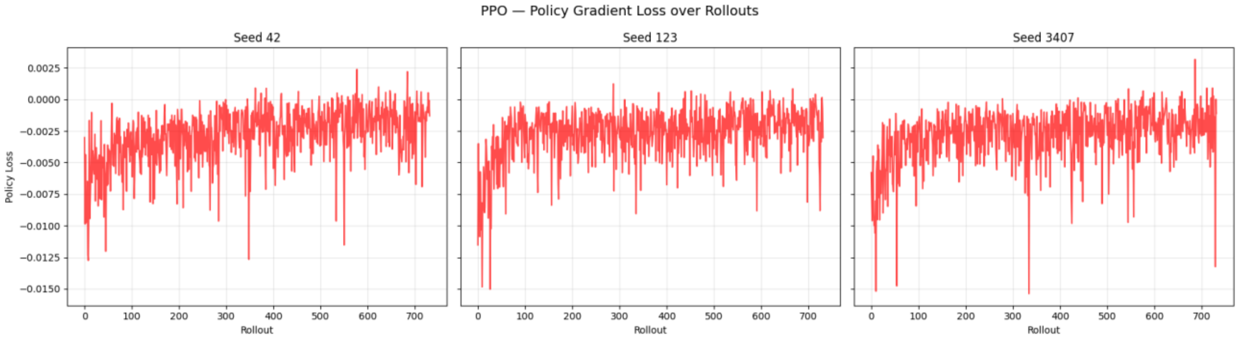


**Figure 11** - PPO policy gradient loss across seeds 42, 123, and 3407.

Figure 11 presents the evolution of the policy gradient loss for PPO across seeds 42, 123, and 3407. The policy loss reflects the magnitude and direction of gradient-based updates applied to the actor network and provides insight into the stability and consistency of policy optimization over time.

Across all seeds, the policy loss exhibits a characteristic pattern: relatively large-magnitude updates during the initial training phase, followed by a gradual reduction in magnitude as training progresses. This behaviour is consistent with the early stage of learning, where the policy undergoes substantial adjustments to move from near-random behaviour towards competent control strategies. As the policy approaches a stable solution, the magnitude of updates decreases, indicating convergence towards a locally optimal policy.

Importantly, the policy loss remains bounded throughout training for all seeds. While sporadic negative spikes are visible, reflecting occasional larger corrective updates, there is no evidence of sustained divergence or instability. This bounded behaviour is consistent with the design of PPO's clipped surrogate objective, which explicitly constrains the magnitude of policy updates to prevent destructive parameter changes. The presence of clipping ensures that even when gradient signals are large, the effective update applied to the policy remains within a trust region, promoting stable learning dynamics.

Minor fluctuations in the policy loss persist throughout training, reflecting ongoing fine-tuning of the policy in response to stochastic transitions and residual variability in advantage estimates. The overall similarity of the loss trajectories across seeds further indicates that PPO's optimization process is robust to random initialization, reinforcing the cross-seed stability observed in reward and episode length curves.

## 10.6 Q-Value Overestimation in DQN Across

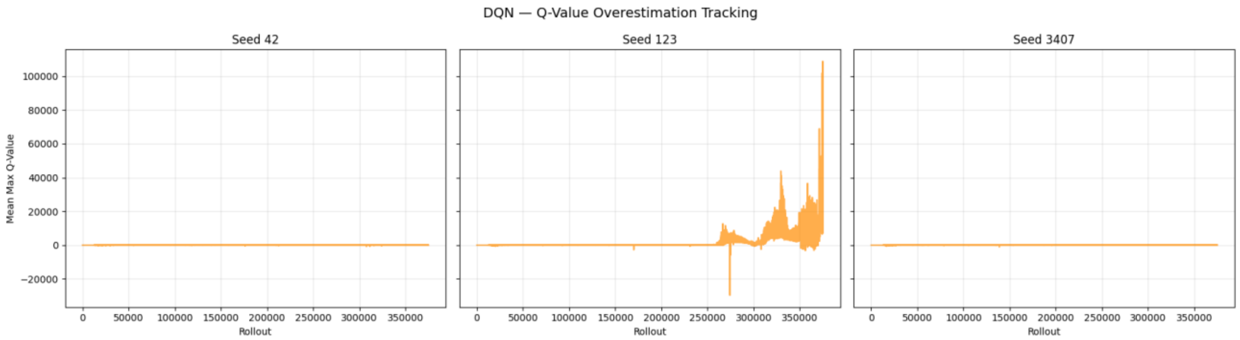


**Figure 12** - DQN mean max Q-value over rollouts for seeds 42, 123, and 3407.

Figure 12 illustrates the evolution of the mean maximum predicted Q-values during training for DQN across seeds 42, 123, and 3407. This diagnostic provides an approximate empirical proxy for tracking overestimation bias in value-based reinforcement learning, where the max operator in the Bellman target can systematically bias Q-value estimates upward.

For seeds 42 and 3407, the mean maximum Q-values remain within a relatively bounded range throughout training, exhibiting only modest growth over time. This suggests that the target network updates and replay-buffer sampling dynamics are effectively constraining runaway overestimation. The stability of Q-value magnitudes in these runs is consistent with the observed stable learning trajectories and sustained high performance after convergence.

In contrast, seed 123 exhibits a pronounced late-stage escalation in the magnitude of predicted Q-values, with sharp spikes and large excursions both above and below the typical range observed in the other seeds. This behaviour coincides temporally with the explosion in value loss and the catastrophic performance collapse observed in the reward and episode-length curves for this seed. The divergence of Q-value magnitudes is indicative of severe overestimation and instability in the bootstrapped target updates, where erroneous value predictions can be amplified through the max operator and propagated across training updates.

These empirical patterns provide direct evidence of the overestimation pathology commonly associated with DQN. While target networks mitigate rapid drift, they do not fully eliminate the risk of runaway overestimation under certain stochastic conditions and replay-buffer dynamics. The instability observed for seed 123 highlights how value overestimation can lead to brittle policies and eventual training collapse, reinforcing the motivation for algorithmic extensions such as Double DQN, clipped targets, or distributional value methods to further stabilize value estimation.

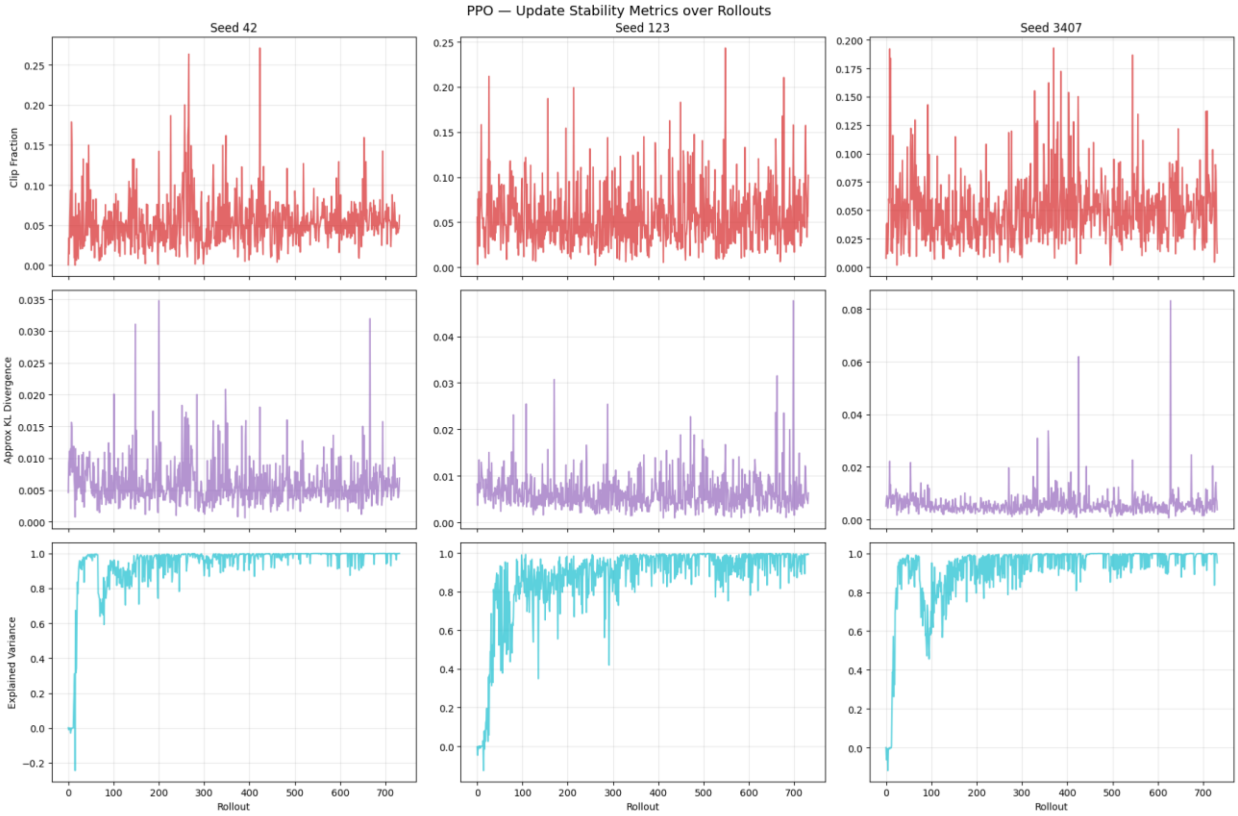


**Figure 13** - PPO clip fraction, KL divergence, and explained variance over rollouts for seeds 42, 123, and 3407.

## 10.7 PPO Update Stability Metrics

Figure 13 reports three complementary PPO-specific stability diagnostics across seeds 42, 123, and 3407: (i) clip fraction, (ii) approximate KL divergence between successive policies, and (iii) explained variance of the value function. Together, these metrics provide a detailed view of how constrained policy updates and critic learning evolve during training.

Across all seeds, the clip fraction remains at moderate levels for most of training, with occasional short-lived spikes. This indicates that PPO's clipping mechanism is actively constraining updates when the policy change becomes too large, while not excessively saturating the objective. The absence of sustained high clip fractions suggests that policy updates remain within the intended trust region for the majority of training, supporting stable learning dynamics.

The approximate KL divergence stays low and relatively stable across seeds, with rare transient peaks. These occasional spikes correspond to moments where the policy undergoes larger adjustments, typically early in training or when adapting to improved value estimates. Importantly, there is no evidence of runaway KL divergence, indicating that PPO's clipped objective effectively prevents destructive policy updates and maintains continuity between successive policies.

The explained variance of the value function increases rapidly from low or negative values during early training to values close to 1.0 for all seeds, indicating that the critic learns to accurately predict returns. High and sustained explained variance throughout training reflects stable and reliable value function approximation, which is crucial for producing low-variance advantage estimates and supporting consistent policy improvement. Minor dips in explained variance correspond to transient mismatches between the critic and rapidly improving policy but are quickly corrected.

Collectively, these stability metrics provide strong empirical evidence that PPO's constrained update mechanism (via clipping) and advantage-based optimization lead to well-behaved training dynamics. The bounded clip fraction and KL divergence indicate that policy updates remain controlled, while the high explained variance demonstrates effective critic learning. These properties help explain the smoother learning curves, absence of catastrophic collapse, and cross-seed robustness observed for PPO in the reward and episode-length analyses.

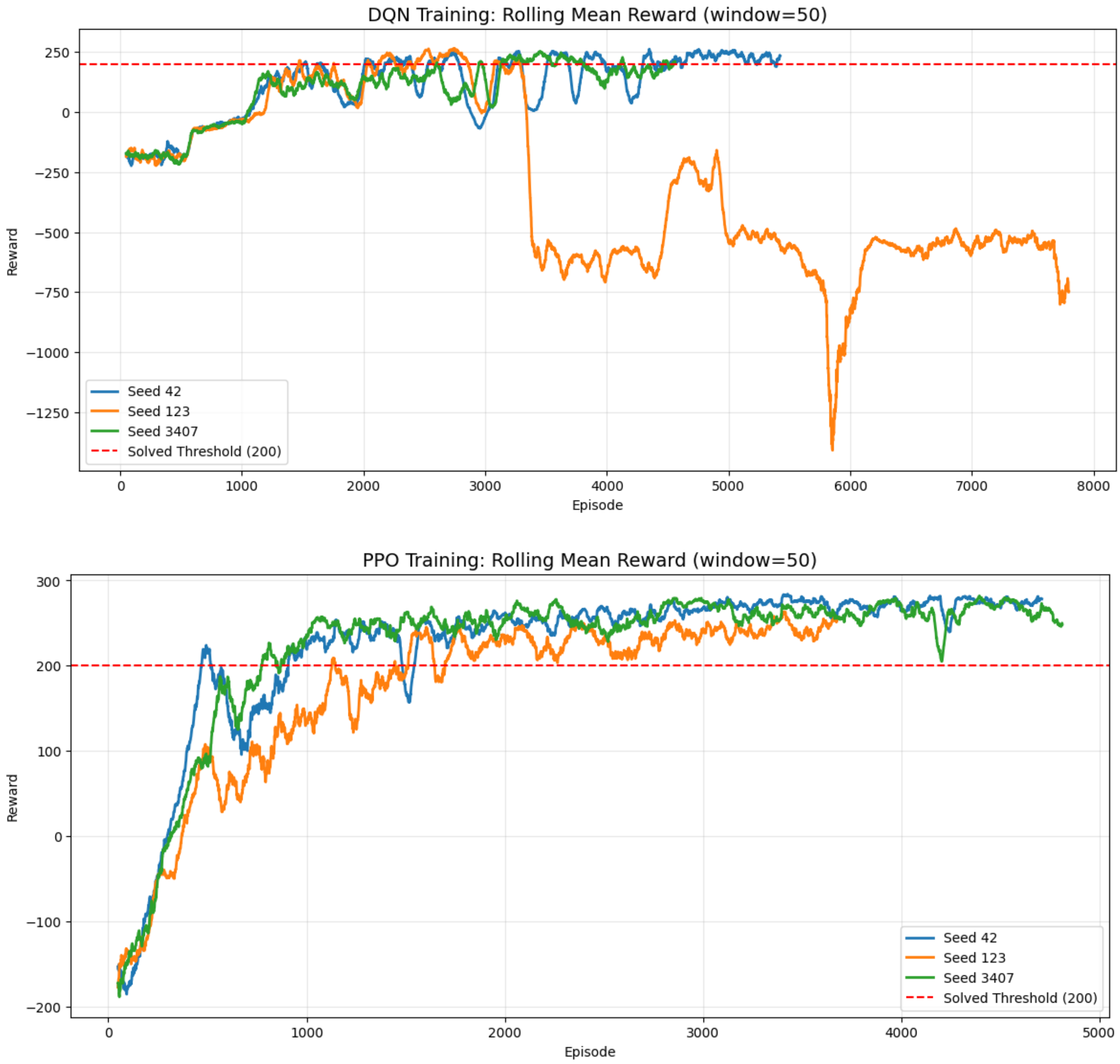


**Figure 14** - Rolling mean reward for DQN (top) and PPO (bottom) across seeds 42, 123, and 3407, with solved threshold at 200.

## 10.8 Cross-Seed Rolling Reward Comparison

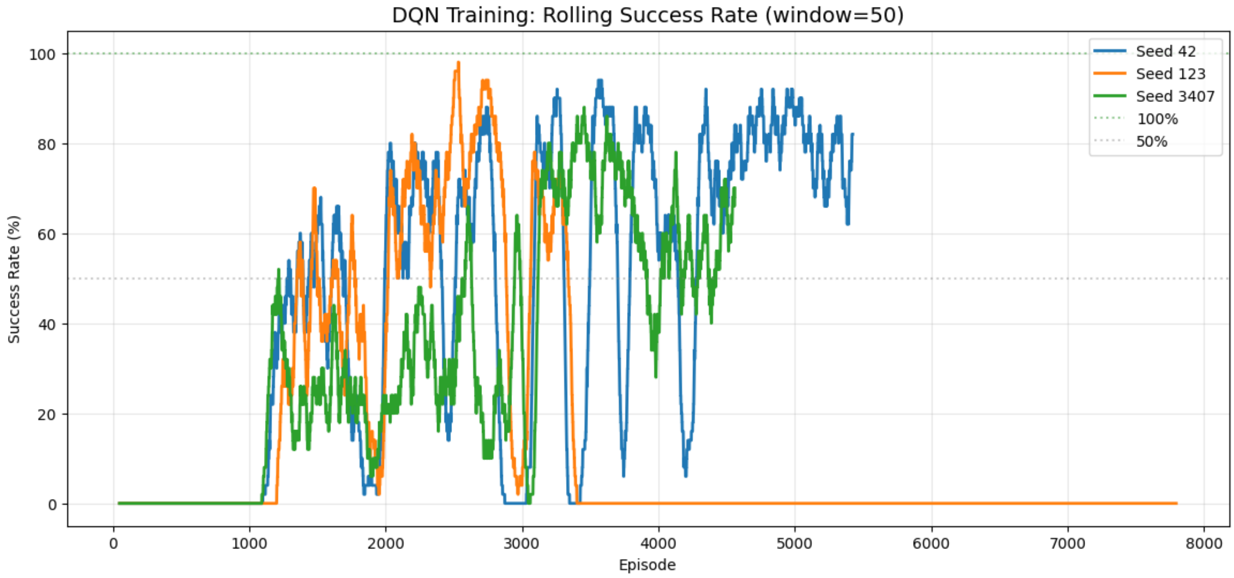
Figure 14 overlay the rolling mean episodic reward (window = 50 episodes) across the three seeds for DQN and PPO, respectively. By smoothing short-term variance, these plots provide a clear view of sample efficiency, convergence speed, and cross-seed stability relative to the solved threshold (mean return ≥ 200).

In the DQN overlay (Figure X), seeds 42 and 3407 follow broadly successful learning trajectories: after an initial low-return regime, both curves rise towards the solved threshold and remain mostly above it in later training, indicating convergence to competent policies. However, the trajectories exhibit multiple oscillations and temporary drops, reflecting the characteristic volatility of bootstrapped value learning.

The most critical behaviour is observed for seed 123. Although the rolling reward initially improves and reaches the solved threshold, it undergoes a dramatic collapse around mid-training and never recovers to sustained competent performance. The rolling mean falls deep into negative values and remains unstable, exhibiting additional severe drops later in training. This pattern confirms that DQN's learning process can become unstable after initially successful learning, reinforcing the earlier evidence from value loss divergence and Q-value explosion. Importantly, this also demonstrates why selecting the best checkpoint via periodic evaluation was essential: high-performing DQN behaviour can be transient and may degrade if training continues.

In contrast, the PPO overlay (Figure Y) shows highly consistent behaviour across seeds. All three seeds rise smoothly from negative returns to the solved region and remain stably above the threshold thereafter. While some seeds converge slightly faster than others, the variability is moderate and no catastrophic collapse is observed. Once PPO crosses the solved threshold, performance remains in a stable high-return regime, typically between ~230 and ~290 in rolling average terms.

The rolling overlays provide one of the clearest empirical contrasts between the algorithms. Both methods can solve LunarLander-v3, but PPO demonstrates substantially stronger robustness across random seeds, achieving stable solved performance in all runs and maintaining that performance once reached. DQN can match PPO's peak performance in favourable runs, but exhibits significantly weaker stability, including clear evidence of collapse under seed 123. This difference aligns with the theoretical expectations: PPO's clipped objective and advantage-based updates tend to produce smoother, safer policy improvements, whereas DQN's bootstrapped targets and max-based value estimation can lead to unstable feedback loops and performance degradation.



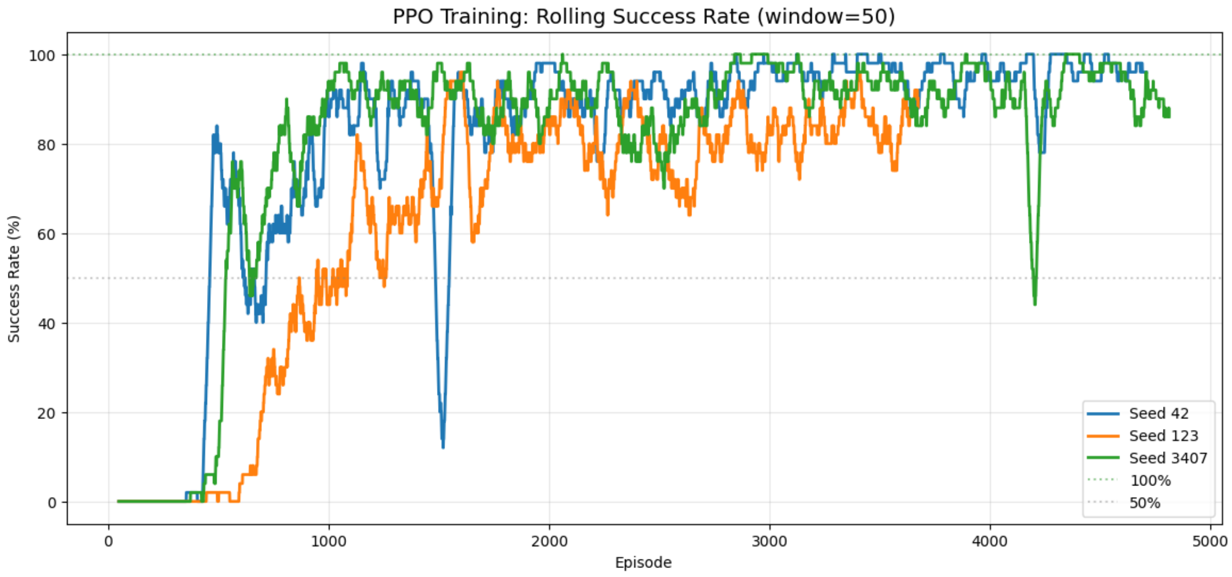


**Figure 15** - Rolling success rate for DQN (top) and PPO (bottom) across seeds 42, 123, and 3407.

## 10.9 Rolling Success Rate

Figure 15 report the rolling success rate (window = 50 episodes) for DQN and PPO, respectively, across seeds 42, 123, and 3407. Success was defined as an episode achieving a total return ≥ 200, consistent with the environment's solved criterion. This metric complements reward curves by translating performance into an interpretable behavioural indicator: the proportion of recent episodes in which the agent reliably executes a competent landing policy.

For DQN, seeds 42 and 3407 show a gradual increase in success rate as learning progresses, eventually reaching high success regimes (often between ~60% and ~90% in the rolling window). However, both seeds exhibit pronounced oscillations, reflecting intermittent failure episodes even after achieving competent behaviour. These oscillations indicate that DQN policies can remain brittle during training, with occasional regressions that reduce short-term reliability.

Seed 123 exhibits a qualitatively different outcome. While the success rate rises significantly during early-to-mid training (approaching very high values at one point), it abruptly collapses and remains near 0% for the remainder of training. This sustained failure regime confirms that the agent no longer produces competent landings and aligns directly with the reward collapse, value loss explosion, and Q-value divergence observed previously. In practice, this strongly reinforces the need for checkpoint-based model selection and early stopping when training DQN, as the final policy after full training may be substantially worse than earlier checkpoints.

For PPO, all seeds show rapid improvement in success rate and converge to consistently high reliability. After the initial learning phase, success rates stabilize predominantly in the ~80%–100% range, indicating that PPO learns not only high-return behaviour but also a robust landing strategy that generalizes across episodes. Occasional transient dips are visible (e.g., isolated drops for seeds 42 and 3407), but these are short-lived and do not lead to persistent collapse. Overall, PPO demonstrates markedly stronger reliability over training compared to DQN.

The rolling success rate plots provide a clear behavioural stability comparison. While mean reward can remain high despite sporadic failures, success rate directly measures how consistently the agent achieves solved-level performance. PPO achieves stronger and more stable success rates across seeds, whereas DQN displays higher volatility and, in the case of seed 123, complete late-stage failure. These findings further support the broader conclusion that PPO provides a more robust training process, while DQN requires careful monitoring and best-checkpoint selection to avoid instability.

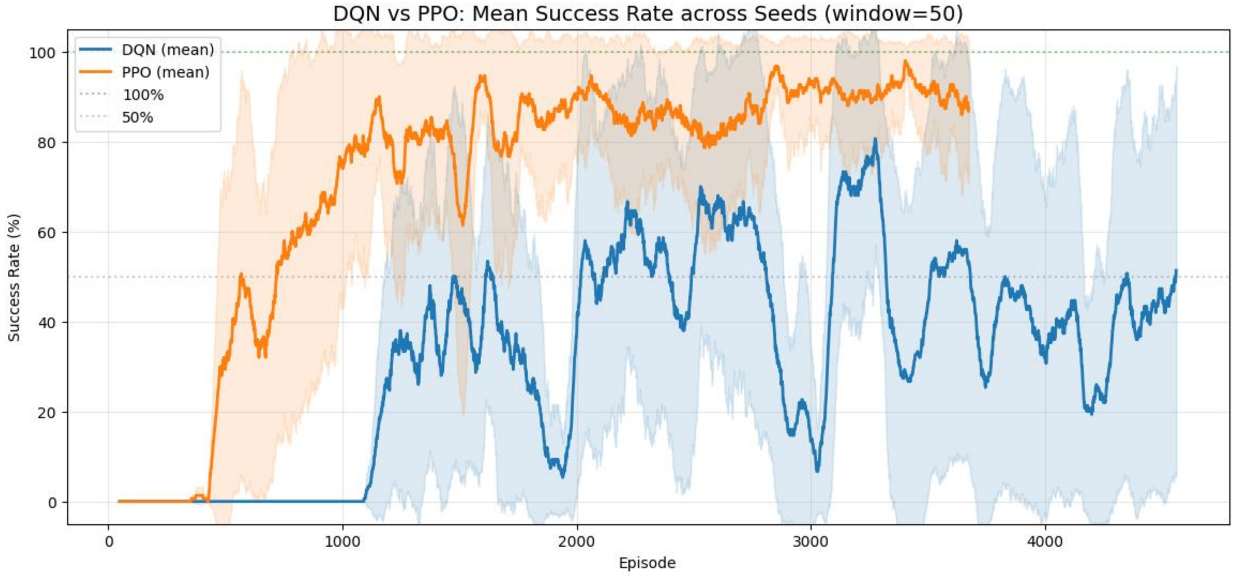


**Figure 16** - Mean success rate for DQN and PPO across seeds.

## 10.10 Cross-Algorithm Comparison: Mean Rolling Success

Figure 16 provides a direct algorithm-level comparison by averaging the rolling success rate (window = 50 episodes) across seeds for DQN and PPO, and displaying mean ± std as a shaded band. This plot compresses seed-level variability into a single view that emphasizes (i) sample efficiency (how quickly each method becomes reliably competent), (ii) robustness (variance across seeds), and (iii) sustained reliability over training.

PPO exhibits a steep and early increase in success rate, reaching high-reliability regimes (≈80–95% rolling success) relatively quickly. After this transition, the mean success rate remains consistently high, and the uncertainty band (± std across seeds) is comparatively narrow. This indicates that PPO not only learns a competent policy earlier, but also does so consistently across random initializations. The stability of the curve around the 90%+ region supports the conclusion that PPO produces robust policies that reliably achieve solved-level episodes throughout training.

DQN increases its success rate later and more gradually. More importantly, the mean trajectory is strongly affected by cross-seed instability, as reflected in the wide shaded band. This wide variance is largely driven by the catastrophic collapse observed in seed 123, which reduces the average success rate and increases dispersion. Even when DQN reaches moderate-to-high success in favourable runs, the overall mean remains unstable and the uncertainty band indicates that performance is far less predictable across seeds than PPO.

This aggregated view reinforces the central comparative message of the project. While both algorithms can achieve solved-level performance, PPO reaches reliable success earlier and maintains it more consistently across seeds. DQN remains significantly more sensitive to stochasticity and initialization, showing larger oscillations and a materially higher risk of late-stage failure. From an applied perspective,where reproducibility and stability matter, PPO demonstrates superior robustness under the controlled experimental setup used in this study.

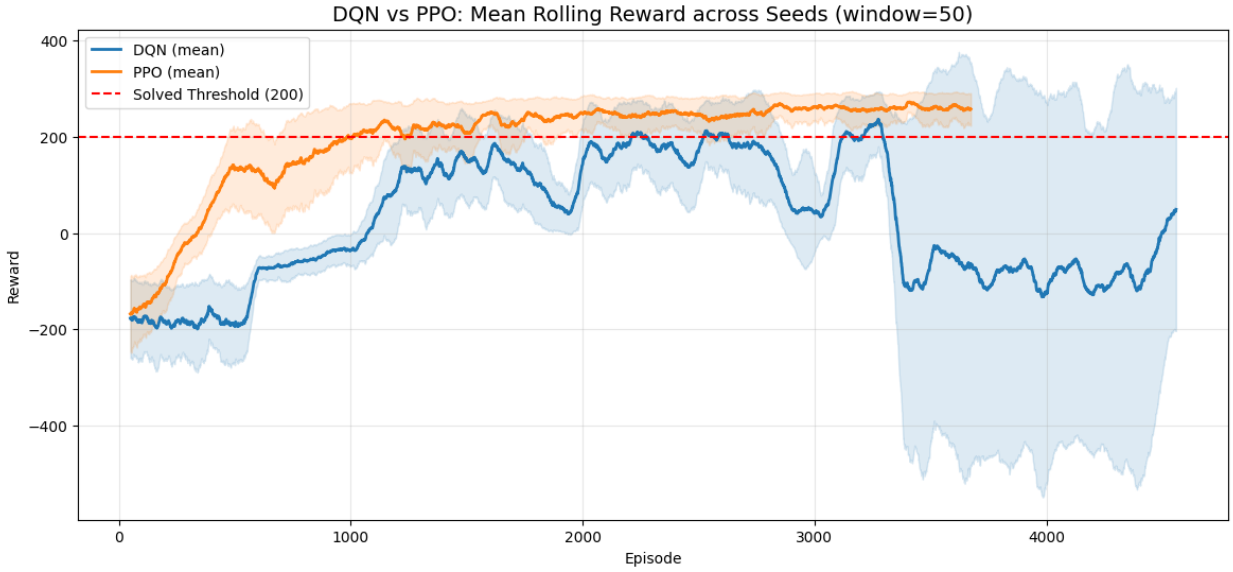


**Figure 17** - Mean rolling reward for DQN and PPO across seeds.

## 10.11 Cross-Algorithm Comparison: Mean Rolling Reward

Figure 17 compares DQN and PPO by plotting the rolling mean episodic reward (window = 50) averaged across seeds, together with a shaded ± std band that captures variability across random initializations. The solved threshold (return ≥ 200) is shown as a reference line, enabling a direct assessment of sample efficiency, convergence, and stability.

PPO shows a rapid increase in mean reward during early training and crosses the solved threshold substantially earlier than DQN. After surpassing the threshold, the mean curve remains consistently above 200, typically stabilizing in the ~240–280 range. The dispersion band around PPO is comparatively narrow, indicating that PPO's learning dynamics are consistent across seeds. This pattern supports the claim that PPO provides both high performance and reliable convergence under the same training budget.

DQN's average reward increases more slowly and crosses the solved threshold later. More importantly, the mean trajectory is far less stable: the curve displays large oscillations and eventually drops well below the solved threshold in the second half of training. The shaded band expands dramatically at later episodes, reflecting substantial cross-seed variability and instability. This widening is consistent with the catastrophic collapse observed in seed

123, which pulls the mean downward and increases variance, even though other seeds maintained competent performance for long segments of training.

This aggregated reward comparison highlights a key trade-off. DQN can achieve high rewards and even match PPO’s peak performance in favourable runs, but its learning process is more fragile and less reproducible across seeds. PPO not only reaches solved performance earlier (higher sample efficiency in terms of episodes) but also maintains it consistently with lower variance, reflecting a more stable optimization process. These results align with theoretical expectations: PPO’s clipped objective constrains destructive policy updates, whereas DQN’s bootstrapped max-based updates can amplify estimation errors and destabilize training under unlucky replay/initialization dynamics.

# 11. Evaluation

The following tables summarize the evaluation performance of each algorithm across all seeds. Each model was evaluated over 20 deterministic episodes. The overall row aggregates all episodes across seeds.

## 11.1 Deterministic Evaluation Summary and Cross-Algorithm Interpretation

Tables 7 and 8 summarize the final deterministic evaluation of the selected best checkpoints for DQN and PPO. Each model was evaluated for 20 deterministic episodes per seed, resulting in 60 episodes per algorithm, and success was defined as return ≥ 200.

| Seed | Mean Reward | Std Dev | Min Reward | Max Reward | Success Rate |
|---|---:|---:|---:|---:|---:|
| 42 | 278.51 | 23.70 | 228.67 | 318.97 | 100.0% |
| 123 | 283.34 | 18.19 | 250.33 | 319.93 | 100.0% |
| 3407 | 261.22 | 25.88 | 215.37 | 310.74 | 100.0% |
| Overall | 274.36 | 24.72 | 215.37 | 319.93 | 100.0% |

**Table 7** — DQN Multi-Seed Deterministic Evaluation Summary (20 episodes per seed)

| Seed | Mean Reward | Std Dev | Min Reward | Max Reward | Success Rate |
|---|---:|---:|---:|---:|---:|
| 42 | 278.43 | 14.56 | 251.45 | 311.70 | 100.0% |
| 123 | 270.83 | 14.06 | 242.53 | 292.55 | 100.0% |
| 3407 | 279.56 | 19.83 | 236.91 | 314.95 | 100.0% |
| Overall | 276.27 | 16.81 | 236.91 | 314.95 | 100.0% |

**Table 8** — PPO Multi-Seed Deterministic Evaluation Summary (20 episodes per seed)

Both algorithms achieved a 100% success rate across all seeds, confirming that the selected checkpoints for both DQN and PPO consistently met the “solved” criterion under deterministic evaluation. In terms of average return aggregated across all evaluation episodes, PPO (276.27) and DQN (274.36) were very close, indicating comparable *final* policy quality once the best checkpoints were selected.

Despite similar mean performance, PPO exhibits lower dispersion in evaluation outcomes overall (Std = 16.81) compared to DQN (Std = 24.72). This suggests that PPO’s final behaviour is more consistent across episodes and seeds, aligning with the training-time evidence of smoother learning and fewer instability events. DQN shows larger variability—most notably driven by higher spread in seeds 42 and 3407—indicating a slightly less uniform performance profile even after best-model selection.

- **DQN:** Seed 123 achieved the highest mean (283.34) among DQN seeds, but DQN also produced the lowest minimum across all its evaluations (215.37, seed 3407). This reflects a wider performance envelope.

- **PPO:** Seed 3407 achieved the highest PPO mean (279.56), while seed 123 was lowest (270.83). Importantly, PPO's minima remain well above the threshold and comparatively tighter.

The deterministic evaluation results must be interpreted together with training curves. DQN's final evaluation is strong because the best checkpoint was selected during training; however, earlier analyses showed that DQN could experience late-stage collapse (notably for seed 123) if training continued unchecked. PPO, in contrast, produced both strong final evaluation and stable learning trajectories, meaning it depended less on "catching" a good checkpoint at the right time.

Under the project's controlled evaluation protocol, both methods produced solved-level agents with excellent deterministic performance. The main differentiator is not final competence but robustness and stability: PPO delivered similar final scores with lower variance and more reliable training behaviour, whereas DQN required stricter monitoring and checkpoint selection to avoid instability.

## 11.2 Evaluation Results (Deterministic Policy Rollouts)

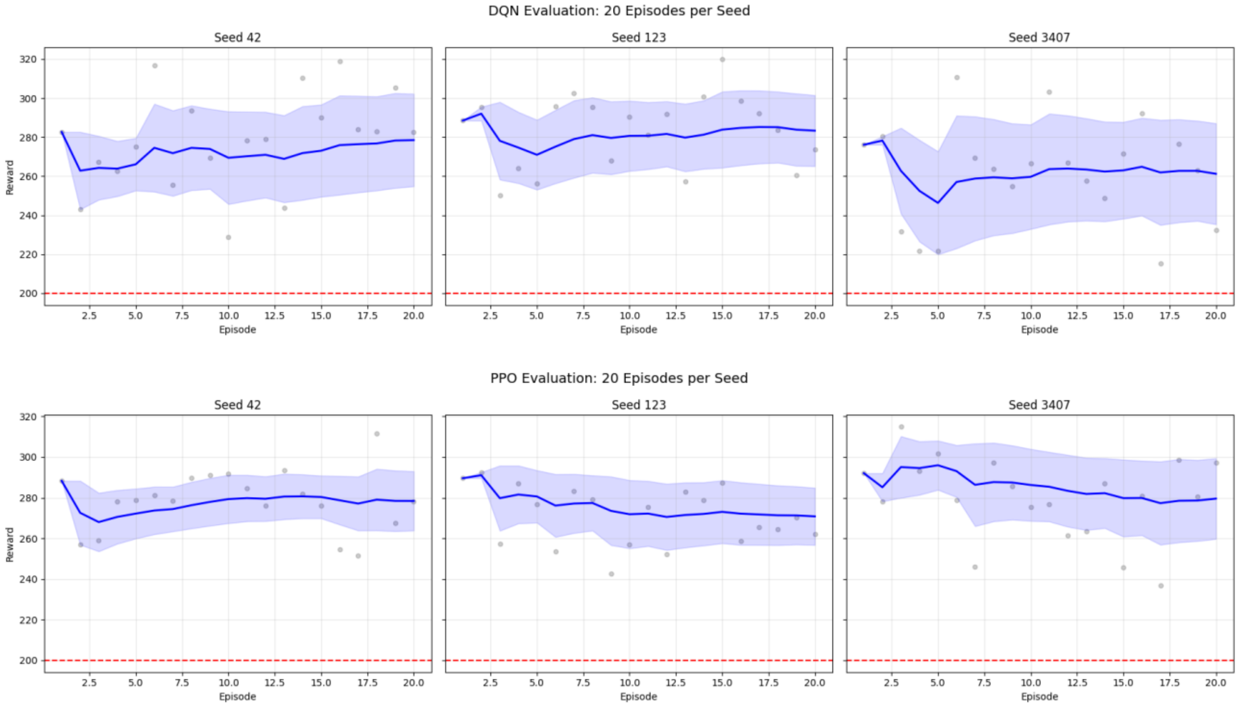


**Figure 18** - Evaluation rewards over 20 episodes for DQN (top) and PPO (bottom) across seeds 42, 123, and 3407.

This section reports the final evaluation performance of DQN and PPO using deterministic action selection over 20 evaluation episodes per seed (seeds 42, 123, and 3407). The objective is to assess final policy quality, robustness, and consistency after training, using the best checkpoint selected by the combined score criterion (mean − std), see Figure 18.

The dashed horizontal line at 200 represents the solved threshold of the LunarLander-v3 environment.

### 11.2.1 DQN Evaluation Performance (20 Episodes per Seed)

Across all three seeds, the DQN agents achieve 100% success rate in evaluation, with episode returns consistently above the solved threshold. However, qualitative inspection of the reward dispersion reveals non-negligible variability across episodes and seeds:

- **Seed 42:** Stable performance around ~270–280 reward, with moderate dispersion. Occasional lower-return episodes remain well above the solved threshold.
- **Seed 123:** Highest mean performance among DQN seeds, but with visible variability across episodes, indicating sensitivity to state-specific trajectories.
- **Seed 3407:** Slightly lower mean return and higher variance compared to the other seeds, reflecting residual instability despite successful convergence.

Although DQN reaches competent final policies, the spread of rewards indicates that value-based learning retains higher stochastic variability at evaluation time, even under deterministic execution. This aligns with earlier training-phase observations of instability and sensitivity to random initialization.

### 11.2.2 PPO Evaluation Performance (20 Episodes per Seed)

The PPO agents also achieve 100% success rate across all seeds, with consistently high returns above the solved threshold. Compared to DQN, PPO exhibits tighter reward distributions and more homogeneous performance across episodes:

- **Seed 42:** High and stable returns (~275–285), with relatively narrow confidence bands.
- **Seed 123:** Slightly lower mean performance than the other PPO seeds, but still consistently above threshold with low variance.
- **Seed 3407:** Strong and stable performance, comparable to seed 42, with limited dispersion.

PPO demonstrates more robust and consistent final policies across seeds. The reduced variance in evaluation rewards suggests that policy-based optimization with clipped updates generalizes more reliably to deterministic execution, yielding stable landing behaviours with fewer outliers.

### 11.2.3 DQN vs PPO Comparative Analysis

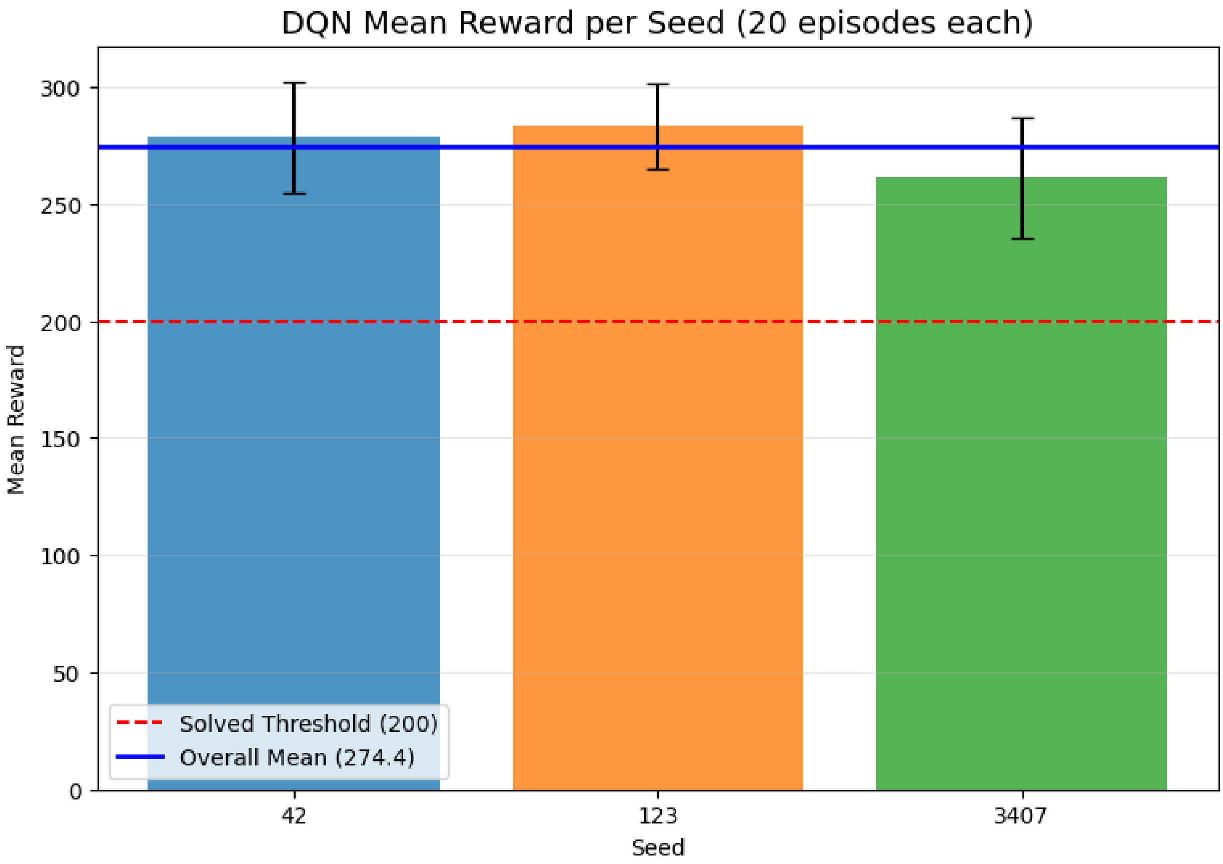
While both algorithms solve the task reliably at evaluation time, qualitative comparison of the evaluation curves highlights clear differences:

- PPO achieves slightly higher or comparable mean returns across seeds relative to DQN.
- PPO shows lower dispersion of episode rewards, indicating more stable and predictable behaviour. DQN, although successful, exhibits greater variability, reflecting residual sensitivity in value estimates.

- PPO performance is more uniform across seeds, whereas DQN shows noticeable seed-dependent differences (particularly in seed 3407).

Both DQN and PPO achieve 100% success rates in deterministic evaluation, confirming that both methods can learn effective landing policies in LunarLander-v3. However, PPO demonstrates superior robustness and consistency in final performance, reinforcing the conclusion drawn from training dynamics that policy-based methods with constrained updates provide more stable and reliable solutions for this continuous-state control problem.

## 11.3 Mean Reward Per Seed



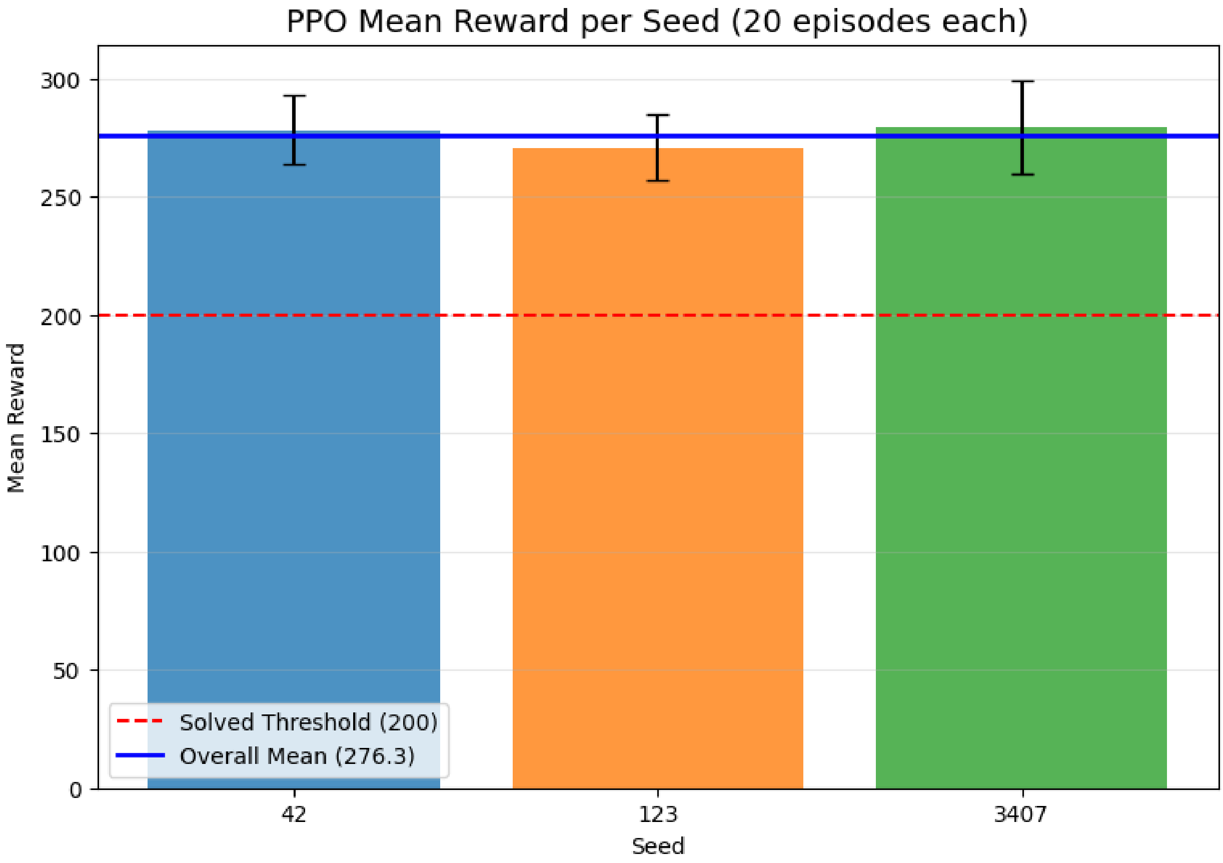
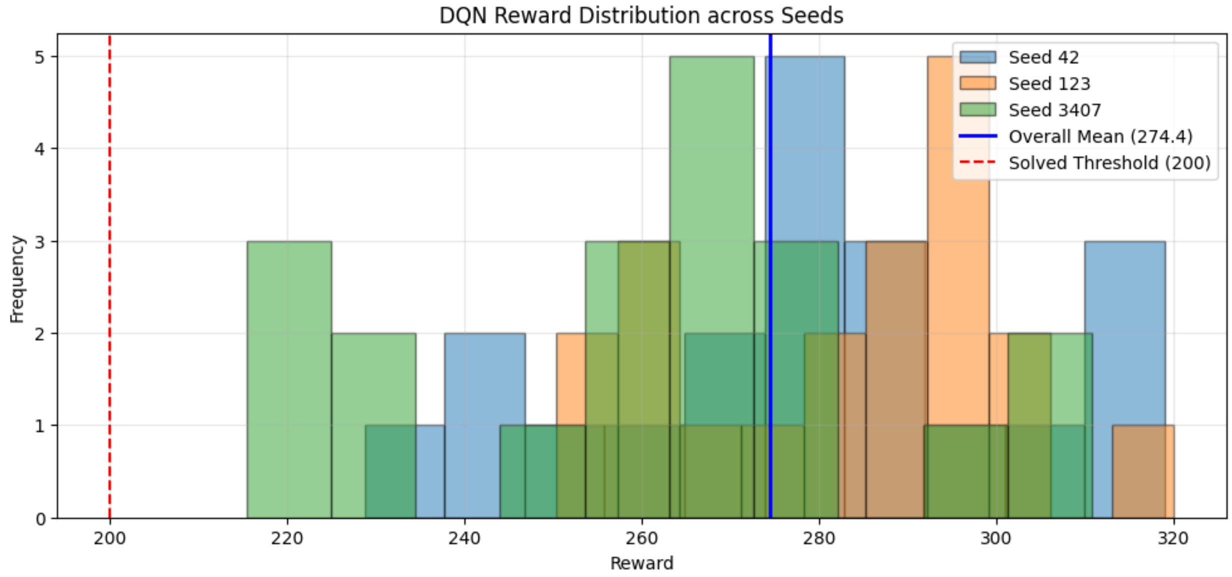


**Figure 19** - Mean evaluation reward per seed for DQN (top) and PPO (bottom), with solved threshold at 200.

Figure 19 shows the evaluation reward per seed for each model. In DQN, the per-seed evaluation plots for DQN show that all selected best checkpoints achieve returns well above the solved threshold (200) across all evaluation episodes. However, the dispersion of points and the width of the confidence band vary by seed. In particular, seed 3407 exhibits larger variability across evaluation episodes, with a few lower-return outliers closer to the threshold. This indicates that while the final DQN policies are competent, their performance is more sensitive to episode-level stochasticity, reflecting residual instability in the learned Q-function.

PPO evaluation curves are consistently above the solved threshold across all seeds, with tighter confidence bands compared to DQN. The per-episode scatter shows fewer low-return outliers and more concentrated performance around the mean. This suggests that PPO's final policies are not only high-performing but also more consistent across evaluation episodes, aligning with the stability observed during training.

## 11.4 Per-Algorithm: Reward Distribution Histograms



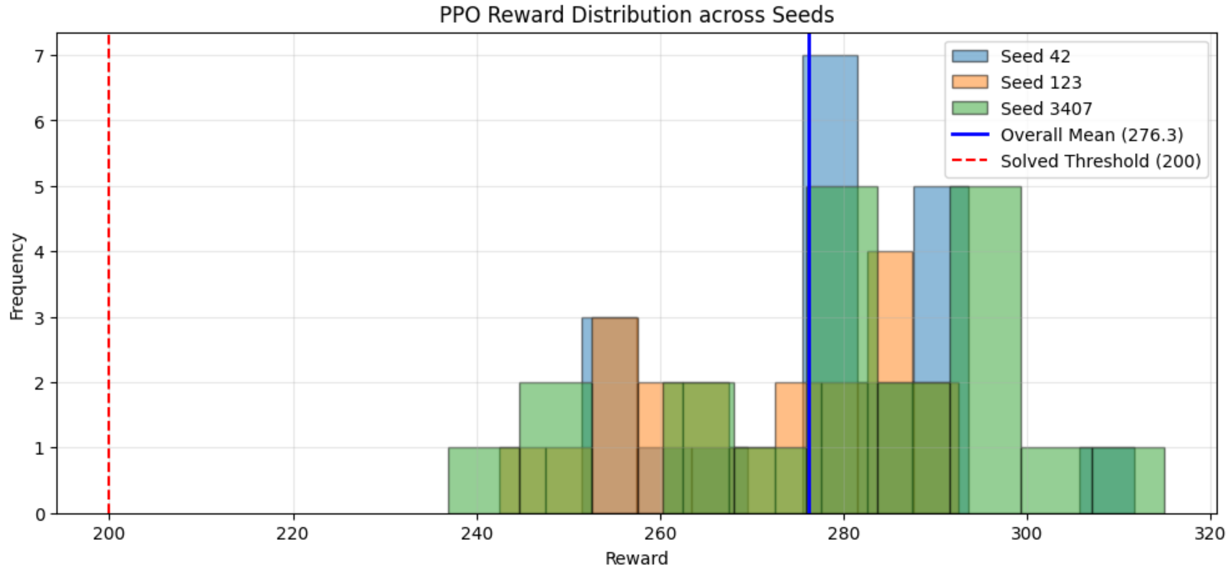


**Figure 20** -Reward distributions for DQN (top) and PPO (bottom) across seeds, with overall mean and solved threshold indicated.

In Figure 20, the reward distributions across seeds illustrate the final performance consistency of both algorithms.

In DQN, the reward distribution shows broader spread across seeds, with noticeable variability between runs. While all seeds are well above the solved threshold (200), the dispersion indicates higher sensitivity to initialization, with some lower-tail mass compared to PPO.

The PPO distribution is more concentrated around higher rewards, with tighter clustering across seeds. This suggests more consistent performance and lower variance in final policy quality.

Both algorithms achieve strong post-training performance, but PPO exhibits a narrower distribution and slightly higher central tendency, indicating more reliable and stable outcomes across random seeds.

# 12. Cross-Algorithm Evaluation

## 12.1 Cross-Algorithm Evaluation – Discussion

| Algorithm | Mean Reward | Std Dev | Min Reward | Max Reward | Success Rate |
|---|---|---|---|---|---|
| DQN | 274.36 | 24.72 | 215.37 | 319.93 | 100.0% |
| PPO | 276.27 | 16.81 | 236.91 | 314.95 | 100.0% |

**Table 9** – Cross-Algorithm Evaluation Summary (60 evaluation episodes per algorithm)

Both algorithms achieve a 100% success rate in the final evaluation (60 episodes per algorithm), confirming that both DQN and PPO are capable of learning effective landing policies for the LunarLander environment. The average returns are very close (≈274 for DQN and ≈276 for PPO), indicating no clear advantage in terms of final mean performance.

However, a relevant difference emerges in terms of performance variability and robustness. PPO exhibits a substantially lower standard deviation (16.81 vs. 24.72 for DQN) and a higher minimum reward (236.91 vs. 215.37), suggesting that PPO produces more consistent and reliable policies across evaluation episodes. In practical terms, this means PPO is less prone to occasional poor landings or suboptimal behaviours at deployment time.

Therefore, although both algorithms reach comparable peak performance, PPO demonstrates superior stability and robustness in execution, reinforcing the conclusions drawn from the training dynamics and stability analyses.

## 12.2 Cross-Algorithm Evaluation: Mean Reward

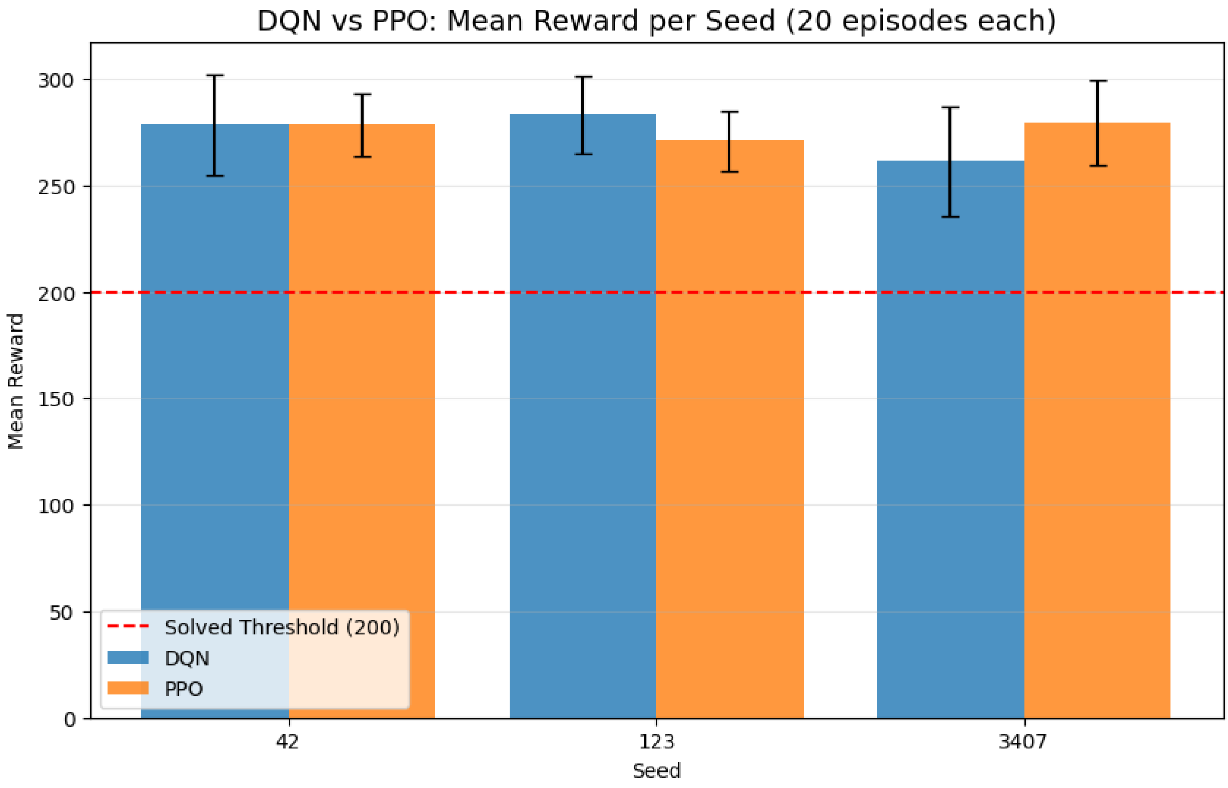


**Figure 21** - Mean evaluation reward per seed for DQN and PPO, with solved threshold at 200.

Figure 21 compares the final evaluation performance of DQN and PPO across the three random seeds (42, 123, 3407), using the mean reward over 20 deterministic evaluation episodes per seed. The dashed red line indicates the solved threshold (reward = 200).

Across all seeds, both algorithms comfortably surpass the solved threshold, confirming that the selected best checkpoints correspond to fully converged policies. However, important differences emerge in terms of cross-seed robustness and relative performance stability:

- **Seed 42:** DQN and PPO achieve very similar mean rewards, with overlapping error bars. This suggests comparable final performance under this initialization, with no clear advantage for either algorithm.
- **Seed 123:** DQN exhibits a slightly higher mean reward than PPO, although the confidence intervals still overlap. This indicates that, for this seed, DQN converged to a marginally better-performing policy, but without strong statistical separation.
- **Seed 3407:** PPO clearly outperforms DQN in terms of mean reward, and the gap is larger than in the other seeds. Moreover, DQN shows wider error bars, pointing to higher variability in evaluation returns and reduced robustness for this particular initialization.

While both DQN and PPO achieve solved performance in all cases, PPO demonstrates more consistent behavior across seeds, whereas DQN shows greater sensitivity to initialization. The performance advantage alternates depending on the seed (DQN slightly stronger for seed 123, PPO stronger for seed 3407), but PPO displays lower variability in the final evaluation stage.

## 12.3 Overall Evaluation: Mean Reward Aggregated

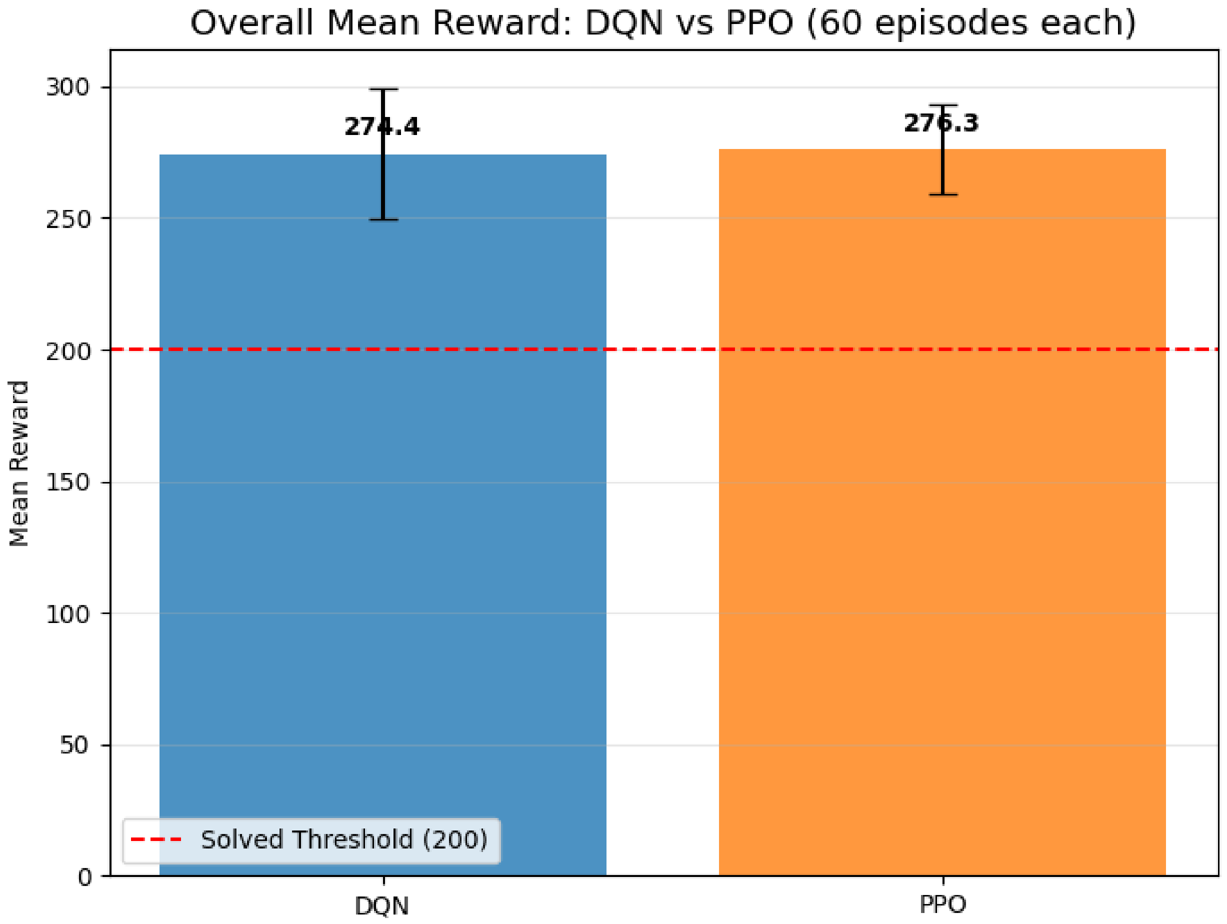


**Figure 22** - Overall mean evaluation reward for DQN and PPO, with solved threshold at 200.

Both algorithms achieve a mean reward substantially above the solved threshold, see Figure 22, confirming that they learned stable and high-performing policies. The overall mean rewards are very close, with PPO achieving a slightly higher average return than DQN:

- **DQN:** Mean reward ≈ 274.4
- **PPO:** Mean reward ≈ 276.3

The difference in average performance is marginal and falls within the range of variability indicated by the error bars, suggesting no statistically significant performance gap under the chosen evaluation protocol. However, when combined with the per-seed analysis (Figure 11.5), this aggregated result highlights two complementary aspects:

- **Peak performance:** Both algorithms are capable of reaching similarly high final performance levels.
- **Robustness:** PPO exhibits slightly lower dispersion in the aggregated evaluation, indicating more consistent behavior across different random seeds.

At convergence, DQN and PPO achieve comparable final performance in terms of mean reward. The small advantage observed for PPO in the aggregated evaluation should be interpreted as a robustness effect rather than a clear superiority in asymptotic performance. This reinforces the conclusion that PPO provides more stable and reliable convergence across random initializations, while DQN can reach equally strong performance in favorable conditions.

## 12.4 DQN vs PPO Evaluation Reward Distribution

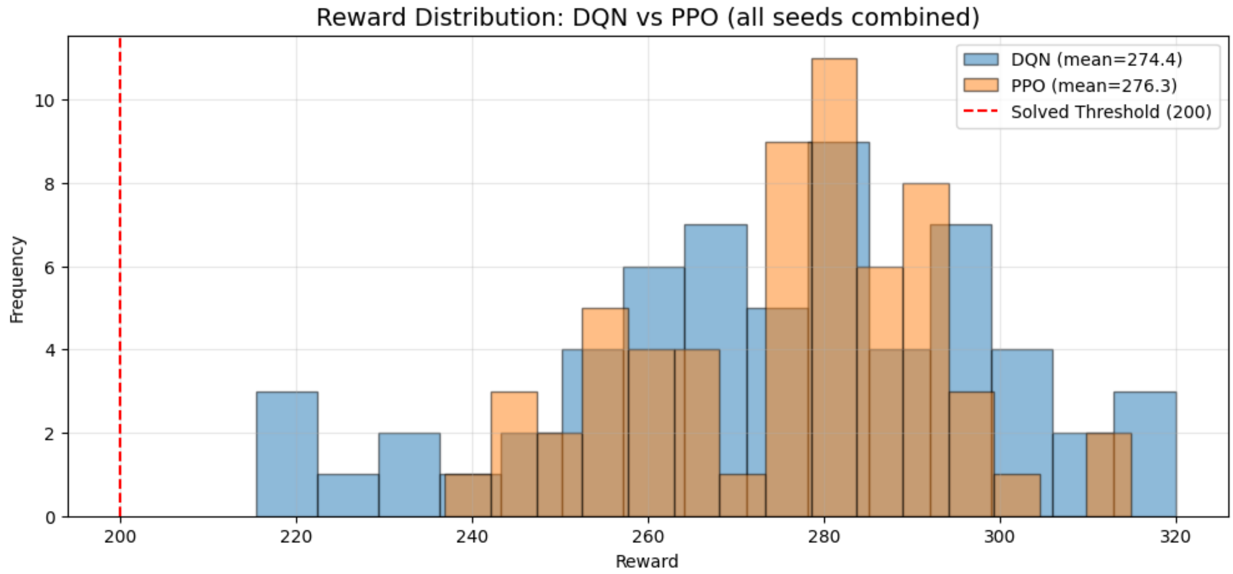


**Figure 23** - Reward distributions for DQN and PPO across all seeds, with overall means and solved threshold indicated.

The distributions of both algorithms are clearly shifted well to the right of the solved threshold see Figure 23, confirming that all evaluated policies consistently operate in the solved regime. Most of the mass of both distributions lies approximately in the range [250, 300], with occasional higher-reward episodes extending above 300.

Several observations can be drawn:

- The distributions largely overlap, which is consistent with the very close mean rewards reported previously (DQN ≈ 274.4, PPO ≈ 276.3). This indicates that both methods achieve comparable average performance at evaluation time.
- PPO exhibits a slightly more concentrated distribution around its mean, whereas DQN shows marginally heavier tails, including both lower outliers (closer to 220–240) and higher peaks (above 300). This suggests that DQN can occasionally achieve very high rewards but with somewhat larger variability.
- No evaluation episode falls below the solved threshold, indicating that both algorithms produce robustly solved policies under deterministic evaluation.

The reward distributions reinforce the conclusions drawn from the mean-based analysis: DQN and PPO reach comparable final performance levels, while PPO displays slightly tighter concentration around its mean, pointing to marginally higher consistency across evaluation episodes. DQN, in contrast, exhibits broader dispersion, reflecting higher variance in episodic outcomes despite similar average performance.

## 12.5 DQN vs PPO Reward Distribution

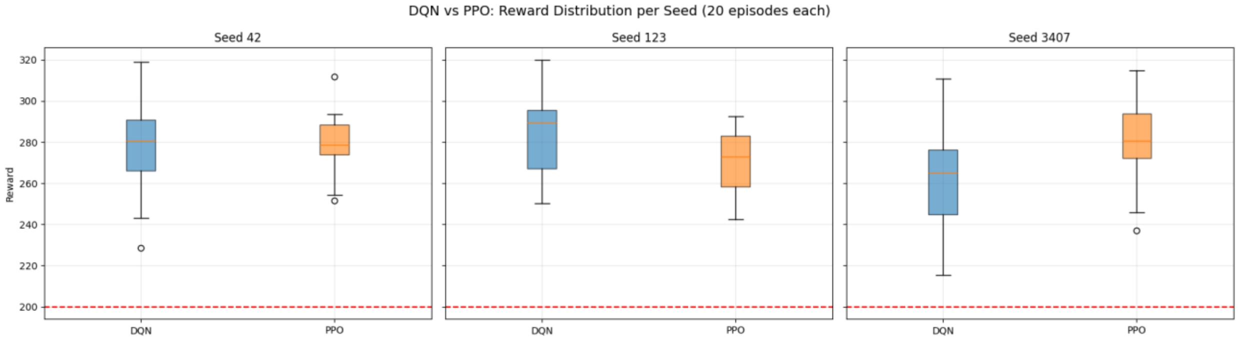


**Figure 24** - Boxplots of evaluation rewards for DQN and PPO across seeds 42, 123, and 3407.

In Figure 24, the per-seed boxplots compare the reward distributions of DQN and PPO over 20 evaluation episodes. PPO shows a tighter interquartile range and fewer low-end outliers across seeds, indicating more consistent performance. DQN achieves comparable or slightly higher medians in some seeds, but with larger dispersion, reflecting higher sensitivity to stochasticity and occasional lower-reward episodes.

# 13. Statistical Significance

Two statistical tests assess whether the observed performance difference between DQN and PPO is significant. The Mann-Whitney U test compares reward distributions (non-parametric, no normality assumption). The Chi-squared test compares success rates as a proportion.

| Metric | DQN Value | PPO Value | Statistical Test | Test Statistic | p-value | Significant ($p < 0.05$) |
|---|---|---|---|---|---|---|
| Mean Reward | 274.36 | 276.27 | Mann–Whitney U | 1771.0 | 0.8811 | No |
| Success Rate (≥ 200) | 100.0% | 100.0% | Chi-squared (not applicable*) | – | 1.0000 | No |

**Table 10** - Sample size per algorithm: 60 evaluation episodes (20 episodes × 3 seeds)

## 13.1 Statistical Interpretation of Evaluation Results

The statistical significance tests indicate that there is no evidence of a meaningful performance difference between DQN and PPO under the evaluation protocol adopted. Although PPO achieved a slightly higher mean reward (276.27) than DQN (274.36), the Mann–Whitney U test yielded a high p-value ($p = 0.8811$), showing that this difference is not statistically significant and can be attributed to sampling variability, see Table 10.

Regarding task success, both algorithms achieved a 100% success rate across all evaluation episodes and seeds. As a consequence, the chi-squared test is not informative in this setting due to the absence of variance in the success outcomes. This result indicates that, once trained, both DQN and PPO consistently solve the environment, and the evaluation phase does not reveal systematic failures for either method.

These findings suggest that performance differences observed during training dynamics do not translate into statistically distinguishable outcomes at evaluation time. In practical terms, both algorithms can be considered equally effective in terms of final policy quality for this environment, with PPO offering advantages mainly in terms of training stability and sample efficiency rather than final asymptotic performance.

# 14. Baseline Comparison

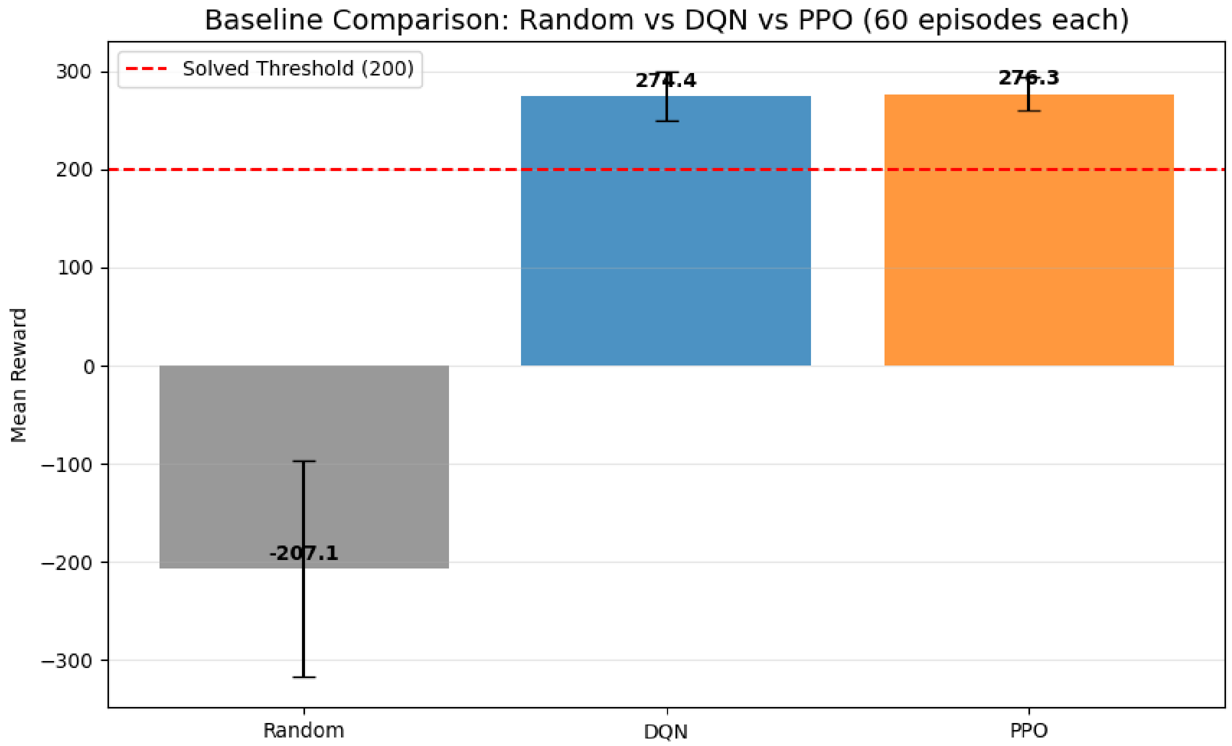


**Figure 25** - Mean reward comparison for Random, DQN, and PPO over 60 episodes, with solved threshold at 200.

| Agent | Mean Reward | Std Dev | Min Reward | Max Reward | Success Rate |
|---|---|---|---|---|---|
| Random | -207.06 | 110.45 | -416.10 | 15.45 | 0.0% |
| DQN | 274.36 | 24.72 | 215.37 | 319.93 | 100.0% |
| PPO | 276.27 | 16.81 | 236.91 | 314.95 | 100.0% |

**Table 11** - Agent Baseline Comparison (Mean, Std, Min, Max)

The baseline comparison highlights a clear and substantial performance gap between the learned policies (DQN and PPO) and the random agent. The random policy yields a strongly negative mean reward (−207.06) with very high variance, failing in all evaluation episodes (0% success rate). This confirms that the task is non-trivial and cannot be solved without learning structured control policies, see Table 11.

In contrast, both DQN and PPO achieve consistently high mean rewards well above the solved threshold (200), with a 100% success rate across all evaluation episodes. This demonstrates that both reinforcement learning methods successfully learned effective landing strategies for the LunarLander environment.

While the average rewards of DQN (274.36) and PPO (276.27) are very similar, PPO exhibits lower variance in performance (standard deviation 16.81 vs 24.72 for DQN), indicating more consistent behaviour across episodes. This suggests that PPO produces more stable policies at evaluation time, even though the final mean performance of both algorithms is comparable.

Overall, this comparison confirms that:

- The improvements over the random baseline are large and unequivocal.
- Both DQN and PPO reliably solve the task.
- PPO offers slightly higher robustness and consistency, whereas DQN achieves comparable peak performance but with higher variability.

# 15. Agent Behavior Analysis

Beyond aggregate performance metrics (reward, success rate and stability), it is important to analyse how the agents behave at a decision-making level. In reinforcement learning, similar performance outcomes can emerge from different control strategies, and understanding these behavioural patterns provides additional insight into the nature of the learned policies.

This section focuses on the action-level behaviour of the trained agents, comparing DQN and PPO in terms of how frequently each action is selected during evaluation. By analysing the distribution of actions (e.g., *Do Nothing*, *Fire Left*, *Fire Main*, *Fire Right*) across all evaluation episodes and seeds, we aim to characterise:

- The degree of conservativeness vs. actuation intensity in each policy
- Potential differences in control smoothness and stability
- Whether distinct action usage patterns may help explain observed differences in learning dynamics, variance, and robustness. DQN: 14,451 total actions collected across 60 episodes PPO: 16,544 total actions collected across 60 episodes

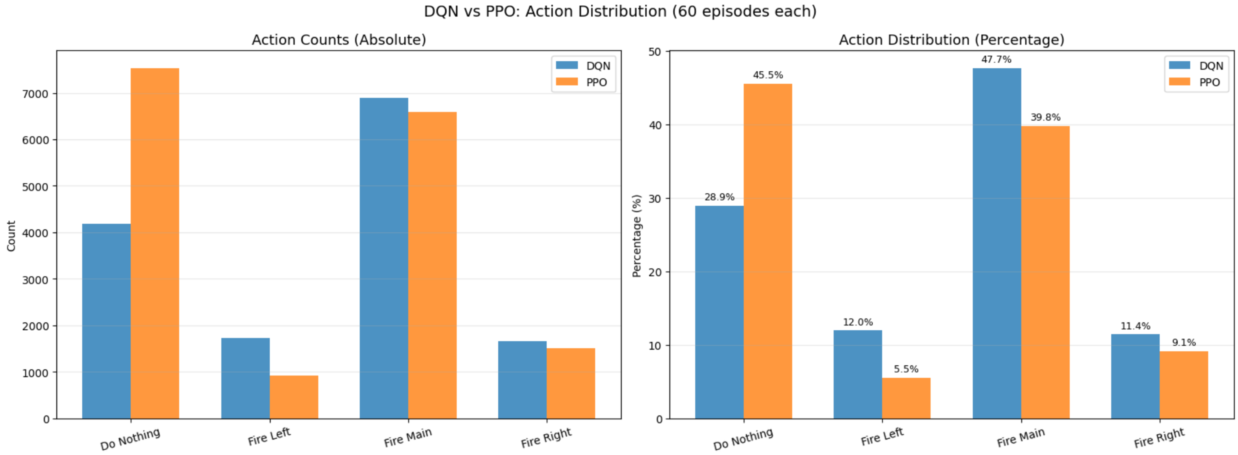


**Figure 26** - Absolute and percentage action distributions for DQN and PPO across 60 episodes.

## 15.1 Action Distribution Analysis

Figure 26 presents the action distribution of the trained agents during evaluation, aggregated across all seeds and episodes. Two complementary views are provided: absolute action counts and relative frequencies (percentages), allowing a direct comparison between DQN and PPO in terms of behavioural tendencies.

Overall, both agents exhibit a dominant preference for the *Fire Main* action, which is consistent with the objective of maintaining continuous thrust to stabilize and control the lander. However, clear behavioural differences emerge in the secondary actions and in the balance between active control and inaction.

PPO displays a substantially higher proportion of *Do Nothing* actions (≈45.5%) compared to DQN (≈28.9%). This suggests that PPO learned a more conservative and smooth control policy, relying more frequently on inaction to maintain stability once a favourable trajectory is achieved. Such behaviour is coherent with PPO’s policy-gradient optimisation and entropy-regularised updates, which tend to promote smoother action distributions and reduced oscillatory control.

In contrast, DQN shows a higher relative usage of corrective actions, particularly *Fire Left* (≈12.0% vs. ≈5.5% for PPO) and *Fire Right* (≈11.4% vs. ≈9.1%). This indicates a more reactive control strategy, where the agent performs frequent fine-grained adjustments to maintain balance. This behavioural pattern is consistent with value-based policies, which may respond more sharply to local state-action value differences.

Despite these behavioural differences, both agents achieve comparable performance in terms of mean reward and success rate (Section 11), indicating that multiple control strategies can lead to similarly optimal outcomes in this environment. Nevertheless, the PPO policy appears to favour smoother and more conservative trajectories, whereas DQN exhibits a more interventionist style with higher actuation frequency.

These findings highlight the importance of behavioural analysis in reinforcement learning evaluation, as performance metrics alone may obscure qualitative differences in how agents interact with the environment and execute control policies.

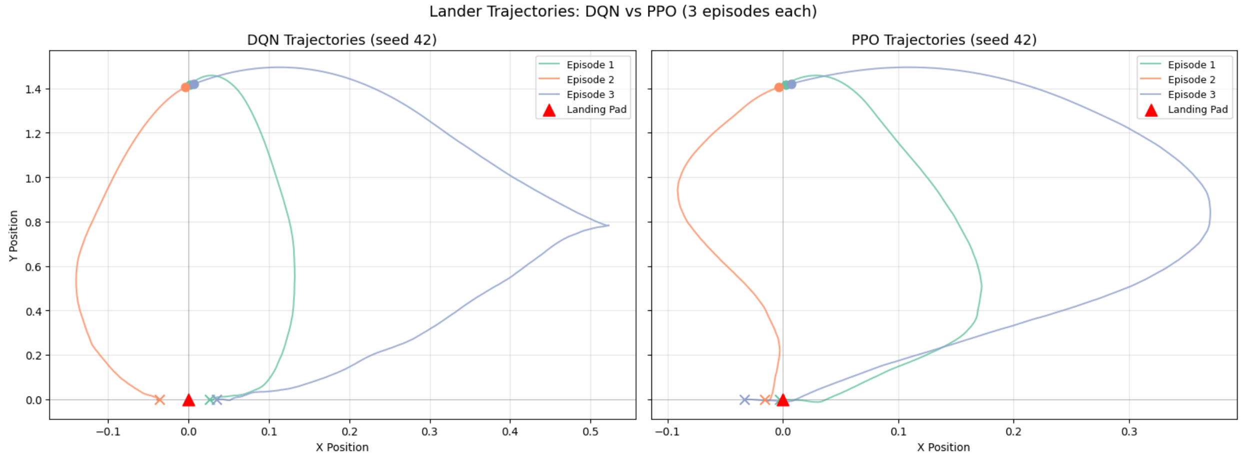


**Figure 27** - Lander trajectories for DQN and PPO (three episodes, seed 42), with landing pad marked in red.

## 15.2 Trajectory Analysis

Figure 27 illustrates representative lander trajectories generated by DQN and PPO over three evaluation episodes (same seed), highlighting qualitative differences in spatial control and descent dynamics. Each curve represents the two-dimensional path followed by the lander from initial descent to touchdown on the landing pad.

Both agents successfully reach the target region, confirming that the learned policies achieve the task objective. However, the geometry of the trajectories reveals distinct control styles.

The DQN trajectories exhibit more pronounced lateral excursions and curved paths, particularly in the horizontal (X) dimension. This indicates a more reactive control strategy, where the agent performs frequent corrective manoeuvres to re-align with the landing pad.

Such behaviour is consistent with value-based decision-making, where local action-value differences may lead to sharper directional adjustments. In some episodes, the DQN agent overshoots laterally before correcting its trajectory, resulting in longer and less direct descent paths.

In contrast, PPO trajectories tend to be smoother and more direct, with fewer abrupt changes in direction. The descent paths show a more continuous and stable convergence towards the landing pad, suggesting that the learned policy emphasises gradual corrections and controlled descent. This behaviour aligns with the policy-gradient optimisation of PPO, which typically yields smoother control policies due to clipped updates and entropy-regularised exploration during training.

These qualitative differences complement the quantitative findings reported in Section 11 and the behavioural statistics discussed in Section 14.1. While both agents achieve comparable performance in terms of final reward and success rate, PPO demonstrates more stable and smooth trajectory profiles, whereas DQN relies on more aggressive corrective actions. This reinforces the interpretation that PPO favours smoother control strategies, while DQN adopts a more interventionist and reactive control style.

Overall, trajectory visualisation provides an intuitive perspective on policy behaviour that is not captured by scalar performance metrics alone, highlighting the importance of combining quantitative evaluation with qualitative behavioural analysis in reinforcement learning studies.

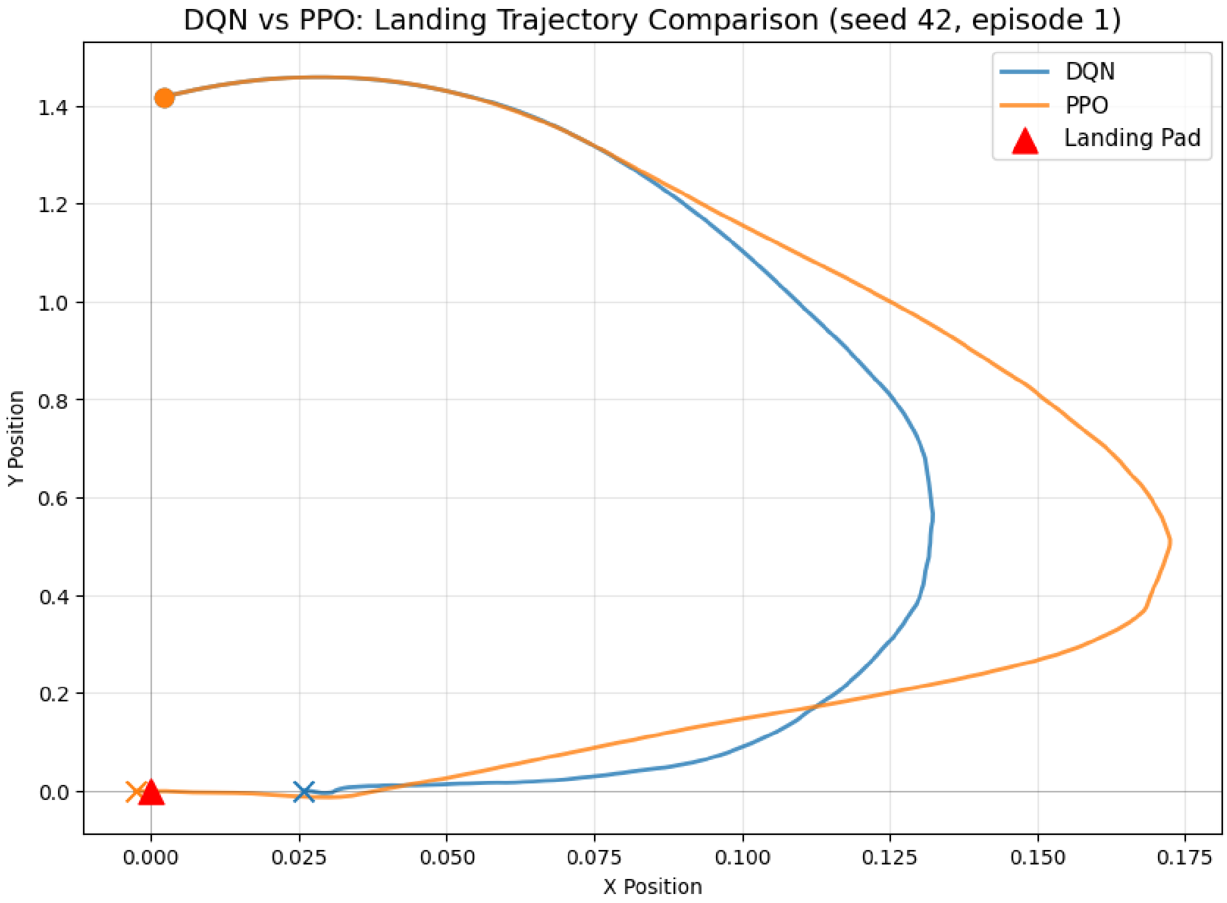


**Figure 28** - Landing trajectories for DQN and PPO (seed 42), with landing pad marked in red.

## 15.3 Direct Trajectory Comparison

Figure 28 presents a direct comparison between DQN and PPO trajectories for the same initial condition (seed 42, episode 1). By overlaying both paths in the same coordinate space, it becomes possible to isolate behavioural differences that are otherwise masked when analysing multiple episodes independently.

Although both agents start from identical initial conditions and reach the landing pad successfully, their control strategies differ visibly. The DQN trajectory exhibits a sharper curvature during the final descent phase, with a more abrupt lateral correction close to the ground. This behaviour reflects a more reactive control pattern, where the agent compensates for positional drift using stronger corrective actions near touchdown.

In contrast, the PPO trajectory shows a smoother and more progressive descent path. The lateral deviation is corrected earlier and more gradually, resulting in a wider but more continuous arc towards the landing pad. This suggests that PPO learns a policy that anticipates the need for alignment earlier in the descent, reducing the magnitude of last-moment corrections.

This single-episode comparison reinforces the qualitative trends observed in the multi-trajectory analysis (Section 14.2). While both algorithms achieve comparable final performance in terms of reward and success rate, PPO demonstrates smoother spatial control, whereas DQN tends to concentrate corrective actions closer to the terminal phase of the landing manoeuvre.

Such qualitative differences are relevant from a control and safety perspective, as smoother trajectories are typically associated with lower control effort and reduced risk of instability in real-world deployment scenarios.

# 16. Sample Efficiency Analysis

Sample efficiency refers to how quickly an agent improves its performance as a function of the number of interactions with the environment. In reinforcement learning settings, this criterion is particularly relevant, as environment steps are often expensive in real-world applications. In this section, the learning efficiency of DQN and PPO is compared in terms of reward accumulation and success rate as training progresses.

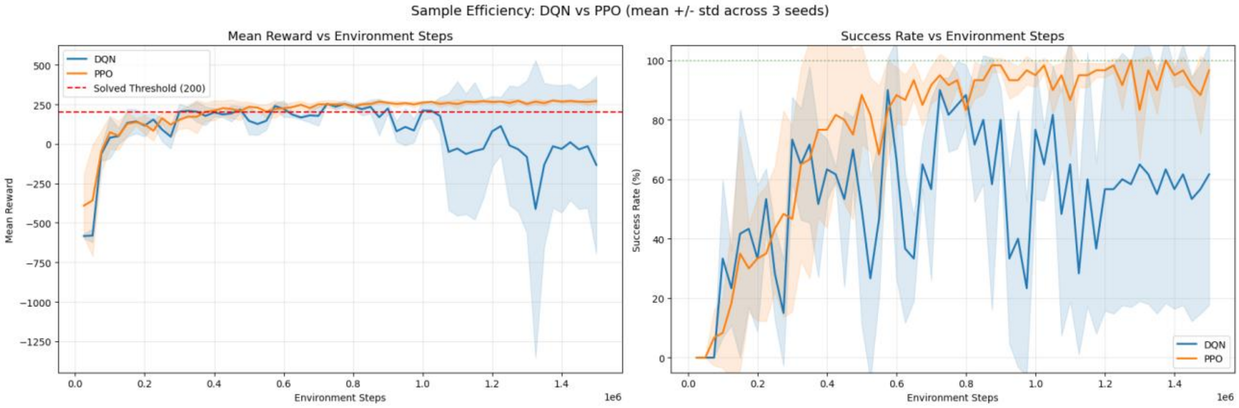


**Figure 29** - Sample efficiency of DQN vs PPO in terms of mean reward and success rate across environment steps.

## 16.1 Mean Reward and Success Rate vs Environment Steps

Figure 29 reports the evolution of the mean episodic reward and the success rate as a function of the total number of environment steps, averaged across three independent random seeds (mean ± standard deviation). The solved threshold (average reward ≥ 200) is indicated as a reference level.

PPO demonstrates substantially higher sample efficiency than DQN. In the early training phase, PPO reaches the solved threshold with fewer environment interactions, showing a steeper initial improvement in both reward and success rate. The success rate curve for PPO rises more smoothly and stabilizes near 100% at earlier stages of training, indicating faster convergence to a consistently successful policy.

In contrast, DQN exhibits slower initial learning and significantly higher variability across seeds, as reflected by the wider confidence bands. Although DQN temporarily reaches high reward and success levels, its performance is less stable, with noticeable drops in both metrics at later stages of training. This behaviour suggests that, while DQN can achieve strong performance, it requires more environment interactions and is more sensitive to training instability.

Overall, these results indicate that PPO is more sample-efficient and robust in the LunarLander environment, achieving reliable performance with fewer environment steps and exhibiting more stable learning dynamics across random initializations.

| Algorithm | Seed | First Solved Step |
|---|---|---|
| DQN | 42 | 300,000 |
| DQN | 123 | 300,000 |

| Algorithm | Seed | First Solved Step |
|---|---|---|
| DQN | 3407 | 150,000 |
| **DQN (Mean)** | — | **250,000** |
| PPO | 42 | 150,000 |
| PPO | 123 | 525,000 |
| PPO | 3407 | 325,000 |
| **PPO (Mean)** | — | **333,333** |

**Table 12** – First timestep at which the solved threshold was reached (mean evaluation reward ≥ 200)

# 17. Training Milestone Timeline

This chart overlays the training reward curves with vertical milestone markers showing when each algorithm reached key performance thresholds. The background curves show the rolling mean reward (window=100 episodes) plotted against cumulative environment steps, reconstructed from the per-episode training logs. Three milestones are marked per algorithm:

- Best model (solid line): the checkpoint selected by the combined metric (mean - std) during training
- First solved (dashed line): first point where the rolling 100-episode mean reward >= 200
- First all-pass (dotted line): first point where all 100 consecutive episodes scored >= 200

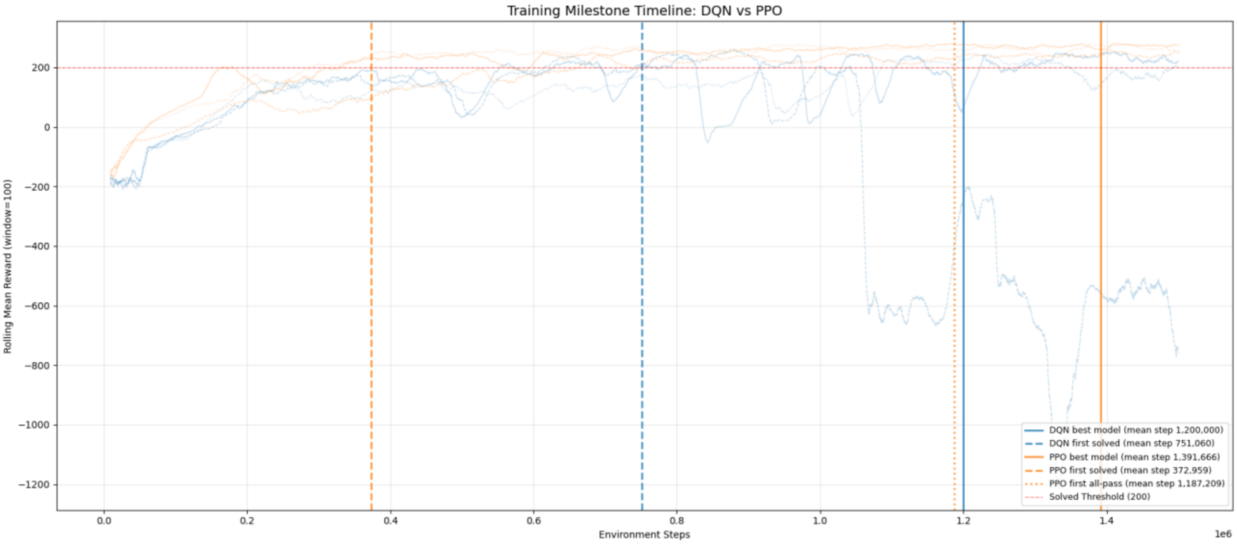


**Figure 30** - Training milestone timeline showing rolling mean reward curves (per-seed, light lines) with vertical markers for best model selection, first solved (rolling 100-episode mean >= 200), and first all-pass (all 100 consecutive episodes >= 200) milestones.

# 18. Final vs Best Model Comparison

| Algorithm | Seed | Best Mean | Best Success | Final Mean | Final Success | Delta Mean |
|---|---|---|---|---|---|---|
| DQN | 42 | 278.51 | 100% | 266.82 | 95% | +11.69 |
| DQN | 123 | 283.34 | 100% | -935.49 | 0% | +1218.83 |
| DQN | 3407 | 261.22 | 100% | 286.92 | 100% | -25.70 |
| PPO | 42 | 278.43 | 100% | 276.39 | 95% | +2.04 |
| PPO | 123 | 270.83 | 100% | 247.72 | 85% | +23.11 |
| PPO | 3407 | 279.56 | 100% | 267.67 | 95% | +11.88 |

**Table 13** - Best model vs final model (20 episodes per evaluation)

Positive Delta Mean = best model selection improved over end-of-training snapshot.

# 19. Hyperparameter Exploration Summary

| Parameter | Value |
|---|---|
| policy | MlpPolicy |
| learning_rate | linear_schedule(...) |
| learning_starts | 50,000 |
| buffer_size | 750,000 |
| batch_size | 128 |
| gamma | 0.99 |
| exploration_fraction | 0.12 |
| exploration_final_eps | 0.1 |
| target_update_interval | 250 |
| train_freq | 4 |
| gradient_steps | 4 |
| policy_kwargs | {'net_arch': [256, 256]} |
| total_timesteps | 1,500,000 |
| device | cpu |

**Table 14** – DQN Final Hyperparameters

| Parameter | Value |
|---|---|
| policy | MlpPolicy |
| learning_rate | 0.00025 |
| n_steps | 2048 |
| batch_size | 64 |
| n_epochs | 10 |
| gamma | 0.999 |
| gae_lambda | 0.95 |
| ent_coef | 0.01 |
| clip_range | 0.2 |
| total_timesteps | 1,500,000 |
| device | cpu |

**Table 15** – PPO Final Hyperparameters

## 20. Future Works

Several extensions can be pursued to deepen and broaden the scope of this study:

- Stability-Enhanced Value-Based Methods: extend the comparison by including improved variants of DQN, such as Double DQN, Dueling DQN, and Prioritized Experience Replay. These methods explicitly address overestimation bias and sample inefficiency, and would allow a more nuanced assessment of whether the observed instabilities of vanilla DQN persist when modern stabilization techniques are applied.
- Comprehensive Hyperparameter Sensitivity Analysis: perform systematic hyperparameter sweeps for both DQN and PPO (e.g., learning rate, network depth and width, target network update frequency, clipping range, entropy coefficient). This would clarify how sensitive each algorithm is to configuration choices and whether PPO's apparent robustness is preserved under less carefully tuned settings.
- Cross-Environment Generalization: replicate the experimental protocol across environments with different dynamics and reward structures, including continuous-control benchmarks (e.g., BipedalWalker, MuJoCo tasks) and sparse-reward settings. This would test the external validity of the conclusions and assess whether PPO's stability advantage generalizes beyond LunarLander.
- Scalability and Compute-Efficiency Trade-offs: analyze wall-clock training time, computational cost, and energy consumption across algorithms and hardware configurations (CPU vs GPU). Such analysis is critical for understanding practical deployment trade-offs, especially in resource-constrained or real-time learning scenarios.
- Robustness Under Distribution Shift and Noise: introduce controlled perturbations to the environment (e.g., sensor noise, altered gravity, reward shaping changes) to evaluate the robustness and adaptability of learned policies. This would provide insights into the resilience of value-based versus policy-gradient methods under non-stationary conditions.
- Long-Horizon and Safety-Constrained Settings: extend the evaluation to tasks with longer horizons and safety constraints, incorporating risk-sensitive or constrained RL formulations. This would help assess whether PPO's stable update mechanism translates into safer learning dynamics compared to DQN in safety-critical domains.
- Representation Learning and Architectural Choices: investigate the impact of alternative network architectures (e.g., deeper networks, residual connections, shared encoders) and auxiliary objectives on learning stability and sample efficiency. Jointly learning state representations may reduce variance and improve convergence properties for both algorithm families.

These directions would contribute to a more comprehensive understanding of the robustness, scalability, and practical applicability of value-based and policy-gradient reinforcement learning methods in diverse real-world settings.

# 21. Conceptual Discussion (Theory–Empirical Connection)

This section provides a critical theoretical discussion of the core conceptual differences between value-based and policy-based reinforcement learning, explicitly grounded in the empirical findings observed in the LunarLander-v3 experiments. Rather than presenting theory in isolation, the discussion links each conceptual aspect to the training dynamics, stability patterns, and performance variability observed for DQN and PPO in this project.

## 21.1 Value-Based vs Policy-Based Learning: Practical Implications

### 21.1.1 Value-Based Learning (DQN)

Value-based methods aim to approximate the optimal action-value function:

$$Q^{\pi}(s,a) = \mathbb{E} * \pi[\sum * t = 0^{\infty} \gamma^{t} r_{t} \mid s_{0} = s, a_{0} = a]$$

and derive the policy implicitly:

$$\pi(s) = \underset{a}{\operatorname{argmax}} Q(s,a).$$

In DQN, a neural network approximates

$$Q(s,a;\theta)$$

, and learning proceeds via bootstrapped Bellman updates:

$$Q(s,a) \leftarrow r + \gamma \max_{a'} Q(s',a').$$

In the experiments, DQN was able to achieve high final performance, occasionally matching or exceeding PPO in peak reward. However, training trajectories were highly non-monotonic, with pronounced oscillations and, in some seeds, late-stage performance collapse. This behaviour directly reflects the structural properties of value-based learning: policy quality is entirely mediated through the accuracy and stability of Q-value estimates. When value estimation becomes unstable, the induced policy degrades, even after the task has already been “solved”.

This explains why strong DQN checkpoints appeared transiently and why model selection and early stopping were critical to recover high-performing policies.

### 21.1.2 Policy-Based Learning (PPO)

Policy-based methods parameterise the policy directly:

$$\pi_{\theta}(a|s)$$

and optimise expected return:

$$J(\theta) = \mathbb{E} * \pi * \theta[R].$$

Using policy gradients:

$$\nabla_{\theta} J(\theta) = \mathbb{E} * \pi * \theta[\nabla_{\theta} \log \pi_{\theta}(a|s) A^{\pi}(s,a)].$$

In contrast to DQN, PPO displayed smoother, more monotonic learning curves and far greater robustness across seeds. Because PPO updates the policy directly, performance is not mediated through a potentially unstable max operator over noisy value estimates. This structural difference explains why PPO training did not exhibit catastrophic collapses and why its learning curves were more predictable and reproducible across random initialisations.

## 21.2 Overestimation and Instability in Value-Based Algorithms

### 21.2.1 Overestimation Bias

The maximisation operator introduces positive bias:

$$\mathbb{E}[\max_i X_i] \geq \max_i \mathbb{E}[X_i].$$

even when

$$X_i$$

are unbiased estimators. In DQN, the same network selects and evaluates actions, amplifying this bias.

In the experiments, rising Q-values and unstable value loss dynamics were observed in some DQN runs, particularly in seeds that later exhibited performance collapse. These effects are consistent with overestimation bias, where inflated Q-values lead the agent to overcommit to suboptimal actions, destabilising learning.

### 21.2.2 The Deadly Triad

DQN combines:

1. Function approximation
2. Bootstrapping
3. Off-policy learning

This “deadly triad” is known to cause divergence and instability in deep RL.

The late-stage collapse observed in one of the DQN seeds is a textbook example of this phenomenon: despite earlier competent performance, training continued to modify value estimates in a destabilising direction. This highlights that convergence in performance does not guarantee convergence in learning dynamics for value-based methods.

## 21.3 PPO Clipping and GAE as Stability Mechanisms

### 21.3.1 PPO Clipped Objective

PPO constrains policy updates via:

$$L^{CLIP}(\theta) = \mathbb{E}[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)].$$

This prevents large, destructive policy updates.

The absence of catastrophic drops in PPO performance and the bounded KL divergence and clipping fraction observed during training confirm the practical effectiveness of this theoretical mechanism. PPO's clipped updates act as a soft trust region, preventing policy collapse even when gradients are noisy.

### 21.3.2 Generalised Advantage Estimation (GAE)

$$A_t^{GAE(\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}.$$

reduces variance while controlling bias.

The smooth improvement of PPO's learning curves and stable value loss trajectories suggest that GAE contributed to lower-variance gradient estimates, improving sample efficiency and stabilising optimisation. This aligns with the observed faster and more reliable convergence of PPO relative to DQN.

## 21.4 ε-Greedy vs Entropy-Driven Exploration

### 21.4.1 ε-Greedy (DQN)

Exploration is externally imposed:

$$\pi(a|s) = \{\text{random} \quad \text{with prob. } \epsilon,\ \underset{a}{\operatorname{argmax}} Q(s,a) \quad \text{otherwise.}$$

Once ε decays, exploration becomes minimal, making DQN prone to premature exploitation and limited recovery from suboptimal behaviours. This rigid exploration schedule may partly explain why some DQN runs failed to recover after performance degradation.

### 21.4.2 Entropy-Regularised Exploration (PPO)

Entropy is incorporated into the objective:

$$L = L^{CLIP} + \beta \mathbb{E}[H(\pi_\theta)].$$

PPO maintained stochasticity in action selection longer into training, which likely contributed to its ability to avoid brittle policies and to recover from suboptimal trajectories. This intrinsic exploration mechanism aligns with the observed robustness and consistency of PPO across seeds.

## 21.5 Critical Synthesis

The empirical results of this project strongly support the theoretical expectations regarding stability in deep reinforcement learning. While DQN can achieve high performance, its reliance on bootstrapped value estimation, maximisation over noisy predictions, and off-policy updates makes it structurally fragile in deep settings. PPO, by directly optimising the policy and constraining updates via clipping and GAE, demonstrates superior robustness and reproducibility.

Crucially, this study highlights that final performance alone is an insufficient metric for evaluating DRL algorithms. Training stability, sensitivity to random seeds, and resilience to optimisation noise are equally important. From both a theoretical and practical perspective, the results reinforce the view that policy-gradient methods such as PPO offer a more reliable foundation for deep reinforcement learning in complex, continuous-state environments.

# 22. Conclusions

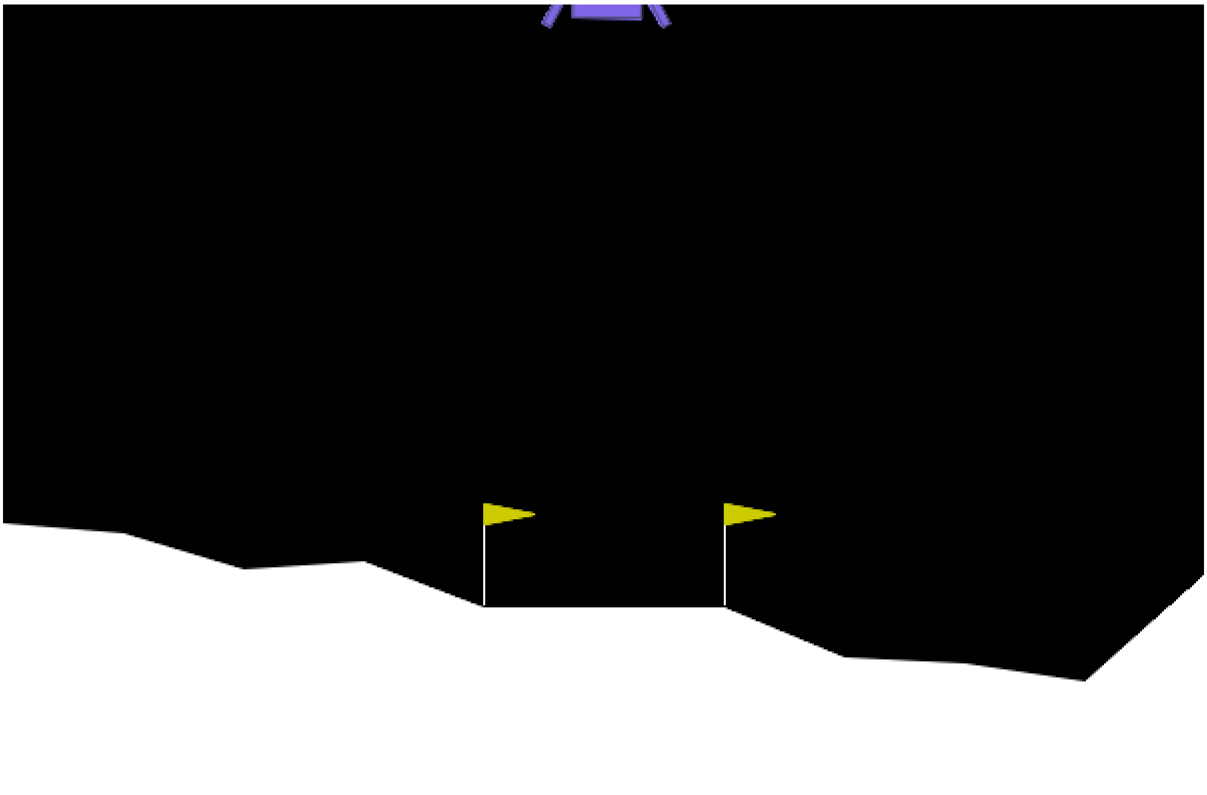
This work presented a systematic empirical comparison between Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO) on the LunarLander-v3 environment, evaluated across multiple random seeds and using a comprehensive set of training and evaluation metrics. Both algorithms were able to solve the task reliably, achieving high mean returns and 100% success rates under deterministic evaluation. These results confirm that, under appropriate configurations, both value-based and policy-gradient methods can learn effective control policies for this benchmark.

Despite comparable final performance, the learning dynamics of the two approaches differed substantially. DQN exhibited pronounced sensitivity to random initialization and training stochasticity, with episodes of instability, late-stage performance collapse in some runs, and evidence of Q-value overestimation. These behaviors highlight the inherent challenges of bootstrapped value estimation and off-policy learning, where divergence can occur even after the task is apparently solved. In contrast, PPO demonstrated smoother and more monotonic learning curves, bounded loss dynamics, and more consistent success rates across seeds, reflecting the stabilizing effect of clipped policy updates and on-policy optimization.

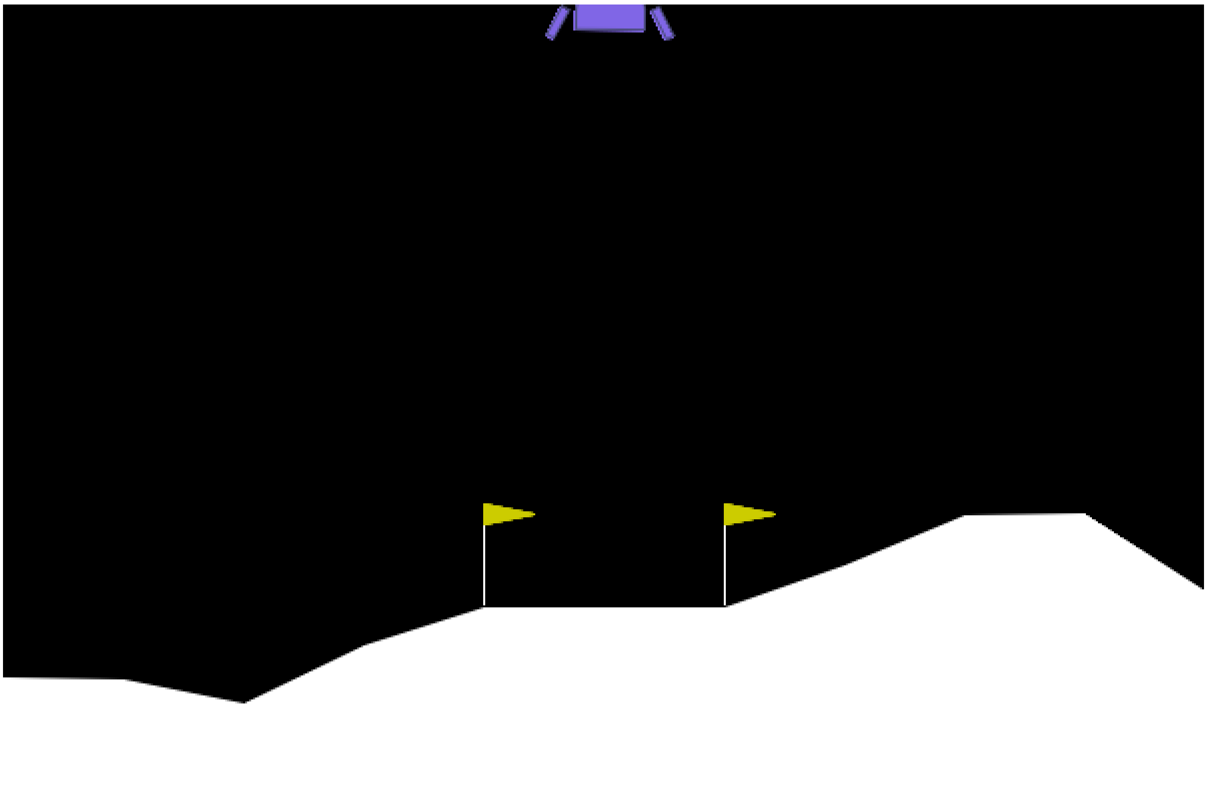
From a practical perspective, PPO showed superior robustness and training stability, as well as more predictable convergence behavior. Although DQN occasionally matched or slightly exceeded PPO in peak performance for specific seeds, this came at the cost of higher variance and reduced reliability over the course of training. The statistical analysis further indicated that the small differences in final average performance between DQN and PPO were not significant, reinforcing the interpretation that stability and robustness—rather than raw peak return—constitute the main differentiating factors between the two methods in this setting.

Generally, the results suggest that PPO is a more dependable choice for practitioners when stability, reproducibility, and consistent convergence are critical requirements. DQN remains competitive in terms of achievable performance but demands careful monitoring, checkpoint selection, and additional stabilization mechanisms to mitigate training instabilities. This comparative analysis underscores the importance of evaluating reinforcement learning algorithms not only by their final scores, but also by their learning dynamics, robustness across seeds, and practical reliability in realistic experimental pipelines.

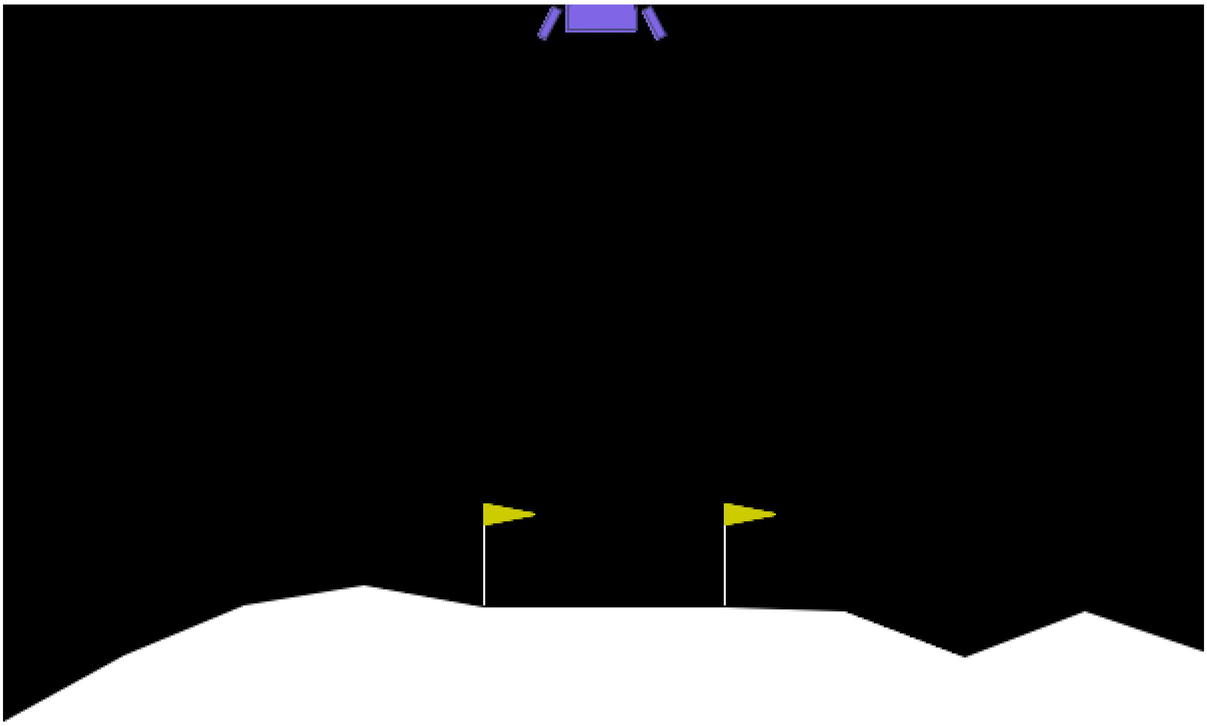
# 23. Lunar Lander Visualizations



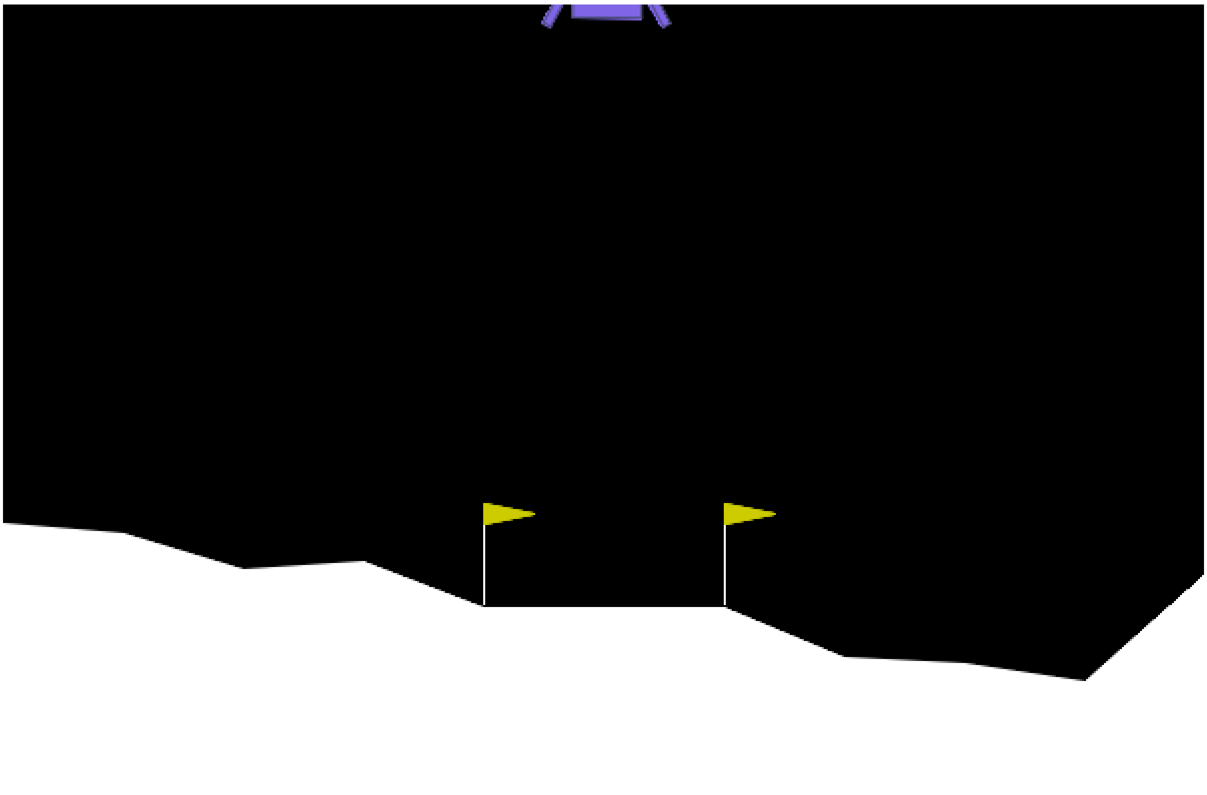
**Gif 1** – DQN landing (Seed 42)



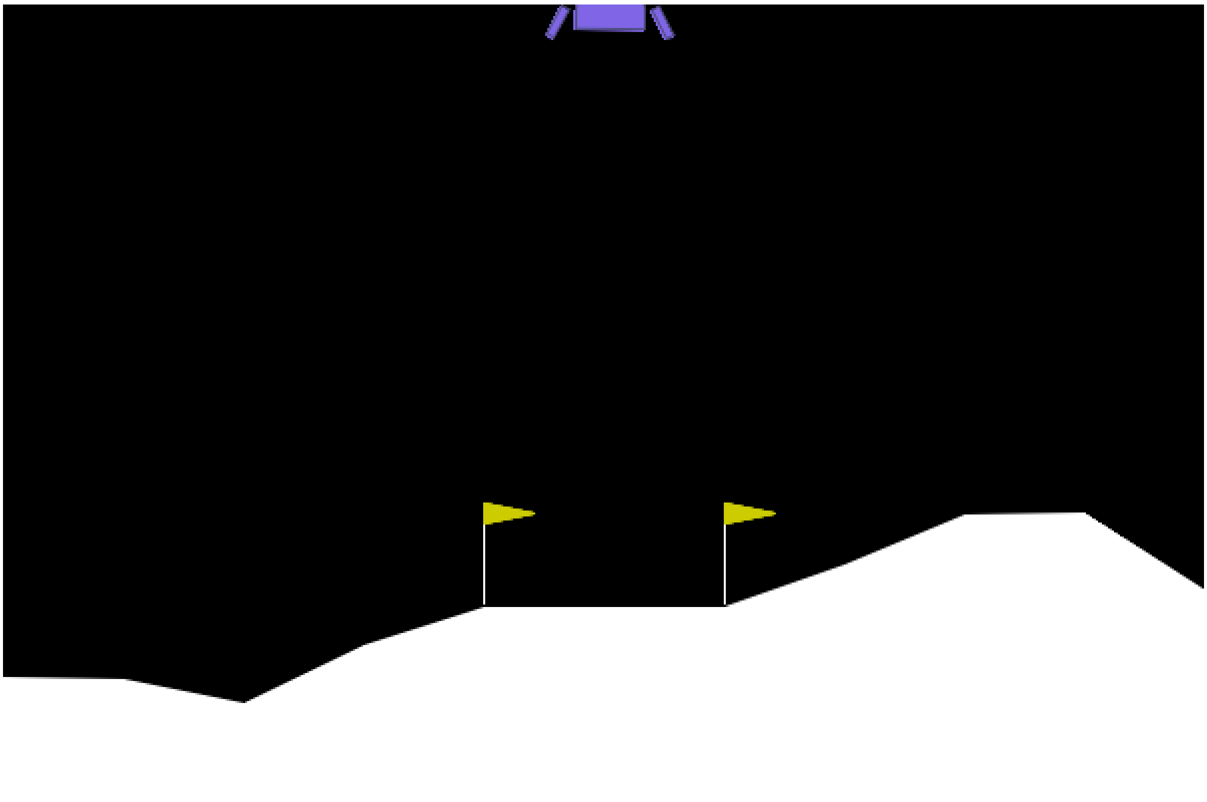
**Gif 2** – DQN landing (Seed 123)



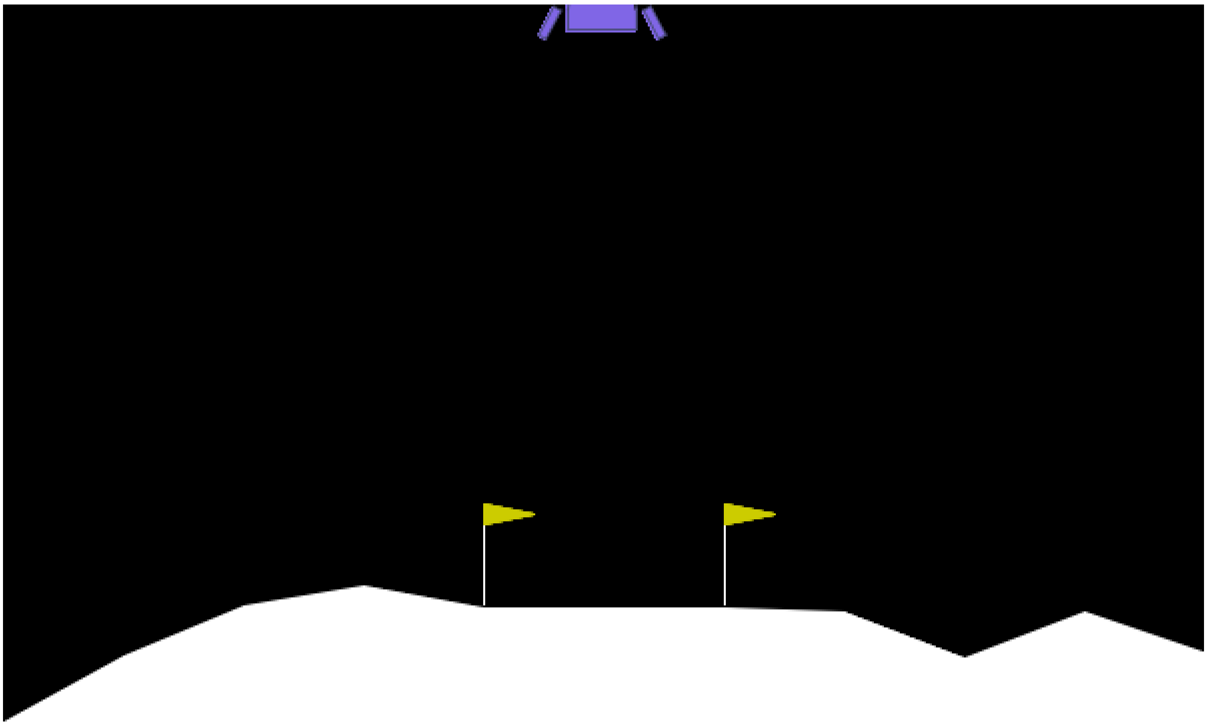
**Gif 3** – DQN landing (Seed 3407)



**Gif 4** – PPO landing (Seed 42)



**Gif 5** – PPO landing (Seed 123)



**Gif 6** – PPO landing (Seed 3407)

# 24. References

- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.
  https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

- Audenis, J. (n.d.). *Neuroevolution Lunar Lander* (GitHub repository).
  https://github.com/josep-audenis/neuroevolution-lunar-lander

- Audenis, J. (n.d.). *Neuroevolution for LunarLander – Project Portfolio*.
  https://josep-audenis.github.io/portfolio/neuroevolutions/

- Gymnasium (Farama Foundation). *LunarLander-v3 environment*.
  https://gymnasium.farama.org/environments/box2d/lunar_lander/

- del Río Ponce, A., Jimenez Bermejo, D., & Serrano Romero, J. (2024).
  *Comparative Analysis of A3C and PPO Algorithms in Reinforcement Learning: A Survey on General Environments*.
  IEEE Access, PP(99), 1–1.
  DOI:10.1109/ACCESS.2024.3472473