Advanced Topics in Deep Learning

2nd Semester - 2025/2026

Mini-Project Assignment – Deep Reinforcement Learning: DQN and PPO in LunarLander-v3

Version 1.0 (2026-02-01)

- **Due date:** March 1st at 23:59 (no late submissions)
- **Group work:** this assignment must be performed in groups of three students.
- **Language:** your project may be written either in Portuguese or in English (preferred).

1. Introduction

Throughout the current module we have introduced the fundamental concepts of Deep Reinforcement Learning, including value functions, policies, exploration strategies, function approximation, and the challenges of stability when learning from interaction. The theoretical classes focused on two major algorithmic families: value-based methods, such as the Deep Q-Network (DQN), and policy-based methods, particularly Actor–Critic approaches and Proximal Policy Optimization (PPO).

In the practical sessions, these concepts were implemented using Gymnasium and Stable-Baselines3, where students learned how to train agents, tune hyperparameters, analyse learning curves, and interpret RL behaviour in real environments.

This mini-project builds directly on both components. By training and comparing DQN and PPO in the LunarLander-v3 environment, students will apply the theoretical principles, reinforce the practical skills gained in the labs, and develop a deeper understanding of how different DRL algorithms behave in practice. The project also invites students to apply critical thinking by proposing and evaluating improvements to each method.

Quick Note: Although PPO is often introduced in this course as a good choice for continuous action spaces, the algorithm fully supports discrete action spaces as well. LunarLander-v3 is therefore a valid environment for both DQN and PPO, enabling a fair comparison between value-based and policy-based reinforcement learning methods.

2. Environment: LunarLander-v3

In the LunarLander-v3 environment, the agent controls a lunar module that must land smoothly between two flags.

The agent receives an 8-dimensional continuous state vector and chooses among 4 discrete actions:

- 0 — Do nothing
- 1 — Fire left engine
- 2 — Fire main engine
- 3 — Fire right engine

The objective is to maximize cumulative reward. Rewards encourage stable landing, penalize excessive fuel consumption, and strongly penalize crashes.

The environment is considered "solved" when the agent reaches an average reward ≥ 200 over 100 consecutive episodes.

3. Tasks

3.1 Agents Implementation and Training

Each group must:

- Train a DQN agent (using Stable-Baselines3).
- Train a PPO agent in the same environment.
- For each algorithm you must test multiple hyperparameter settings. A configuration means changing relevant hyperparameters (e.g., learning rate, batch size, γ, exploration/entropy, network architecture, …).
- For a fair and reproducible comparison, in the end:
  - Train both agents with the same total number of environment steps.
  - Use at least 3 random seeds and report mean ± std.
  - Report the number of gradient updates performed by each agent.
  - Evaluate each final agent over 20 deterministic episodes per seed.
- Save learning curves, hyperparameters, and eventually relevant logs.

3.2 Mandatory Comparative Analysis

The report must include:

- Environment analysis: State space, action space, reward function, termination rules
- Experimental setup
  - Total environment steps used
  - Hyperparameter tables for DQN and PPO
  - Minimum 3 seeds
  - Evaluation: 20 deterministic episodes per seed
- Learning curves
  - Mean return vs. environment steps (mean ± std)
  - Episode length vs. environment steps
  - Exploration metrics: epsilon (DQN) / entropy (PPO)
  - Loss curves: Q-loss (DQN), policy/value/entropy losses (PPO)

    - Optional additional analyses are welcome.
- Comparison of sample efficiency (how quickly each method learns)
- Training stability
    - Variance across seeds
    - Oscillations, divergence, plateaus
    - Overestimation (DQN) and update stability (PPO)
- Qualitative assessment of agent behaviour (videos/gifs)
- A conceptually discussion based on theory:
    - Value-based vs policy-based learning,
    - Overestimation and instability in algorithms
    - PPO clipping and GAE,
    - ε-greedy vs entropy-driven exploration.

In earlier projects, we emphasized experimentation and interpretation of learning behaviour, and high final scores were not required. Here, achieving good performance is part of the goal. You should aim to reach a clearly competent landing policy (at least for one of the agents) and explain how your hyperparameter choices and improvements contributed to it.

4. Objectives

By completing this mini-project, students should be able to:

- Apply the core concepts of Deep Reinforcement Learning in a practical exercise.
- Train and compare DQN and PPO on the same control task, understanding their key differences.
- Run controlled experiments and learn the influence of the several available choices.
- Analyse learning curves and agent behaviour to identify strengths, weaknesses, and failure modes.
- Propose and evaluate improvements to each algorithm based on theory and empirical evidence.
- Present results clearly through a clean notebook and a concise written report.

5. Steps & Hints

The following steps are suggested as a guideline for structuring your work. You do not need to follow them strictly, but they may help you organise the project:

- Set up the environment
    - Make sure you can run the LunarLander-v3 environment from Gymnasium.
    - Run a few random episodes and inspect the states, rewards, and termination conditions to understand the task.
    - Document what you have learned about the environment
- Implement a baseline DQN agent
    - Start with a standard DQN configuration from SB3.
    - Verify whether the agent is able to learn something better than random, even if the learning is unstable or slower.

    - Save the learning curves and record the hyperparameters used.
- Implement a baseline PPO agent
    - Train a PPO agent on the same environment, using the same number of environment steps as for DQN.
    - Compare its behaviour to DQN: learning speed, stability, and ability to reach higher rewards.
    - Again, save learning curves and note all hyperparameters.
- Explore improvements
    - After obtaining baseline results for both agents, you must explore ways to improve their performance and stability.
    - Improvements may involve tuning key hyperparameters (learning rate, batch size, γ, exploration/entropy settings, network architecture, etc.) or enabling specific algorithmic enhancements provided by SB3.
    - You are encouraged to check the official Stable-Baselines3 documentation to understand the available options and their potential impact. In the practical classes we introduced part of the existing hyperparameters, but there are others that you may explore and experiment with.
    - Document the experiments, explaining what you changed, why you changed it, and how it affected training behaviour and final performance.
- Summarize
    - Present the best approach for each of the agents
    - Compare from a critical perspective
    - Conclude with the best approach for the challenge

These are suggested steps and hints, but don't forget to comply with the section 3. Tasks.

6. Deliverables

Your submission should include:

- An introduction to the problem, including Environment description — state space, action space, reward function.
- A Jupyter Notebook with your implementations and experiments.
- Experimental setup — seeds, hyperparameters, methodology
- Results — plots, tables, analysis
- A short report in pdf format including conclusions on the best approaches, why, summary of the learning, limitations, etc…
- Submit also the final trained models for the best DQN and PPO configurations (one model per algorithm).

Note: Writing this report does not replace the need for commenting on your code. You should add inline explanations, observations and reflections directly in the notebook, so that your reasoning process is clear during experiments.
If you prefer, you can also deliver your analysis (short report) in the notebook. But it should be presented in text, eventually with support images, in a text format.

7. Evaluation

Your work will be evaluated based on:

- Success achieved by the trained agents
- Completeness of the experiments performed.
- Clarity of plots and results.
- Depth of analysis in the written report.
- Correctness and organization of the code.

8. Requirements and Submission Guidelines

Please note the following requirements for your submission:

- The project must be carried out in groups of 3 students.
- Submission deadline: 2-Mar at 23:59.
- The project report may be written either in Portuguese or in English (English is preferred).
- Your notebook and report must be well organized and follow the good practices, including:
  - Use of clear sections and structure.
  - Integration of code with explanations and commentary.
  - Compliance with coding conventions and readability standards.
  - Ensuring reproducibility of experiments.
  - Proper handling of imports and installation of dependencies (if required).
- Ensuring reproducibility is always a good idea: fixed seeds, library requirements, clear instructions.
- Submit through Moodle: notebook, report and any other required files

9. Remarks

- You can copy code from the web, but
  - always provide the reference of the code that you copy
  - Make sure that you explain the used code so that another student of the course understands it and can use it
- During the development stage of your work, you should perform as many tests as you wish. However, you must submit a clean version of your notebook, containing only the relevant steps. In order to do so, before submitting you should copy your development notebook into a new file and clean it by deleting spurious tests and code.

- It is not always necessary to have a full-blown analysis, but you are expected to think! For each step, ask yourself why this step makes sense and then explain this in your notebook. Inline explanations in the notebook are required — your reasoning must be visible.

10. Submission Instructions

Your notebook should be submitted through the Moodle platform. You can make several submissions, taking into account that a submission replaces the previous one.
If the data can be easily retrieved for the provided address, and was not subject of change during the assignment, just provide the link to the data in your assignment notebook. Otherwise, submit the data together with your work.

11. Policies

Students may share and/or exchange ideas with each other about the assignments and/or solving them. However, the work turned in must correspond to the individual effort of each group. The following situations are considered fraud:

- Partially copied work
- Facilitating copying by sharing files
- Using other people's material without mentioning the source.

In case of detection of any type of fraud, the work in question will not be evaluated, and will be sent to the Pedagogical Commission or the Pedagogical Council, depending on the gravity of the situation, which will decide the sanction to be applied to the students involved. Plagiarism tools will be used for automatic copy detection.