UAB

Universitat Autònoma de Barcelona

Universitat Autònoma de Barcelona (UAB)

Bachelor's Degree in Artificial Intelligence

PROJECT

Deep Reinforcement Learning

November 7, 2025

Contents

1 Introduction

Some relevant remarks about the project:

- The project will be developed throughout the semester, during which time students will address a **problem of considerable complexity**.
- Projects will be completed in **groups of two or three students**. These groups must remain consistent throughout the project and will be responsible for self-managing tasks such as role distribution, work planning, task assignment, resource management, and conflict resolution.
- Each group will **work independently** to develop their project.
- The project requires students to collaboratively design a **comprehensive solution** to the assigned challenge.
- In addition, students must demonstrate teamwork skills and **present their results to the class**.
- Each project will be evaluated based on a **deliverable** and an **oral presentation** to the class. Active participation in both the preparation of the deliverable and the presentation is essential to receive a project grade.

Punctuation:

- **Part 1**: 3 points
- **Part 2**: 2 points (+1 extra point)
- **Part 3**: 5 points

2 Part 1: Solving the Pong Environment

The Arcade Learning Environment[1] (ALE), commonly referred to as Atari, is a framework that allows researchers and hobbyists to develop AI agents for Atari 2600 roms. It is built on top of the Atari 2600 emulator Stella and separates the details of emulation from agent design. Users can interact with the games through the Gymnasium API[2].

The Pong environment[3] belongs to the Atari suite of environments. In this game, the agent controls the right paddle, while the left paddle is operated by the computer. The objective is to prevent the ball from entering your own goal while attempting to score by sending it past your opponent's paddle.

In this section, you are required to apply the knowledge acquired throughout the course to complete the following tasks:

1. Using the `PongNoFrameskip-v4` environment, apply the necessary **preprocessing steps** to prepare the environment for training. Provide a detailed explanation and justification of all preprocessing steps (or wrappers) used, and clearly specify the resulting observation space after preprocessing.

2. Select **two or more deep reinforcement learning (DRL) models** and compare their performance in the Pong environment.

   Note that you are **not allowed** to use pre-built implementations or third-party code libraries for this task. However, you may develop your own code and reuse material from the course, including examples, activities, or publicly available reference implementations found online.

Remarks:

- You must use **Gymnasium version 1.0.0 or higher** for this task. Previous versions of Gymnasium or the deprecated OpenAI Gym library are not

[1] https://ale.farama.org/
[2] https://gymnasium.farama.org/
[3] https://ale.farama.org/environments/pong/

permitted.

Questions:

1. Describe the **preprocessing steps and wrappers** applied in your experiment, specifying the resulting observation space.

2. Provide a detailed **description of the selected agents** or models.

3. **Train** the models, applying fine-tuning or hyperparameter optimization as needed.

4. **Save** the best-performing models and produce **visualizations** of their behavior and learning curves.

5. **Evaluate** the trained models within the environment, reporting the **average success rate**.

6. **Discuss** the obtained results, analyze the findings, and **justify** the selection of the best model for this environment.

3 Part 2: Pong World Tournament

The PettingZoo[4] is a simple, pythonic interface capable of representing **general multi-agent reinforcement learning** (MARL) problems. PettingZoo includes a wide variety of reference environments, helpful utilities, and tools to create your own custom environments. This activity uses the **AEC API**, which supports sequential turn-based environments. PettingZoo could be combined with Stable Baselines 3[5] or any other library to train the models.

The following libraries are required to properly run this activity:

```
> pip install swig
> pip install box2d-py
> pip install gymnasium
> pip install "gymnasium[atari,accept-rom-license]"
> pip install "stable-baselines3[extra]"
> pip install "pettingzoo[all]"
> pip install supersuit
```

Notes:

- **AutoROM** must be installed on the Python environment and then execute "AutoROM" from the command line to install it.

  ```
  > pip install "autorom[accept-rom-license]"
  > AutoROM
  ```

The primary goal of this section is to train an agent to play the Pong game in a multi-agent environment, adhering to the rules outlined earlier (see details https://pettingzoo.farama.org/environments/atari/pong/).

For your reference, three accompanying notebooks are provided to assist with this task:

1. 20241_Project PML_Part 3: Contains foundational and preliminary information on the environment.

[4] https://pettingzoo.farama.org/index.html
[5] https://stable-baselines3.readthedocs.io/en/master/

2. 20241_Project PML_Part 3 (Sup material 1): Provides specific guidance on training an agent using the SB3 library.
3. 20241_Project PML_Part 3 (Sup material 2): Details the steps for loading a trained agent to play in a Gymnasium single-agent environment.

Questions:

1. **Select** one RL model to play this game.
2. Tune the **parameters** to optimize the agent:
   (a) Indicate the range of parameters and the results obtained for each one.
   (b) Compare them and select and justify your decision.
3. **Train** the agent with these parameters:
   (a) Report the results of your agent in a single-player environment (for instance, Pong[6] from Gymnasium or Gym).
4. **Export** your trained agent to a file:
   (a) Save and load[7] from SB3 could be used.
   (b) Or include the code needed to load and run your trained agent.
5. **Playing Pong**: Using the model (or models) trained in the previous exercise, answer the following questions:
   (a) Test your model (or models) in 100 episodes of Pong (using the *PettingZoo* environment). Report the results (win rates, rewards, etc).
   (b) Export one video (*mp4* format) of a whole episode.

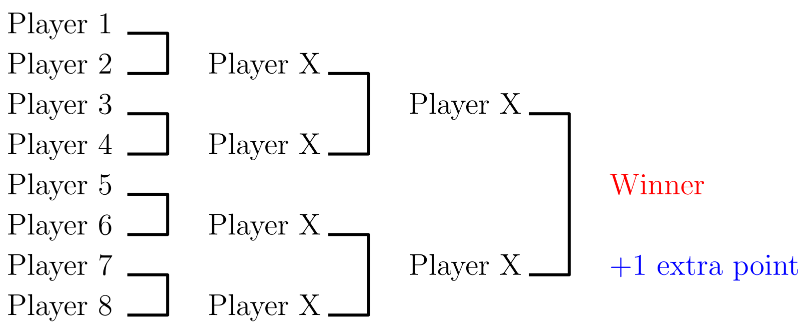
[6] https://gymnasium.farama.org/environments/atari/pong/
[7] https://stable-baselines3.readthedocs.io/en/master/guide/save_format.html

Pong World Tournament

We are excited to announce the Pong World Tournament, where all groups that have trained an agent to play Pong in a multi-agent environment using PettingZoo will compete!

Tournament rules:

- Players will be randomly assigned to a slot (Player 1 to Player 8).



- Each match will consist of 5 independent games, with players alternating sides of the board.
- The first player to win 3 or more games will be declared the winner.
- Winners will advance to the next phase and face new opponents.
- The winner of the tournament will earn **1 extra point**.

4 Part 3: Solving a "complex" ALE environment

In this section, you are required to apply the knowledge gained from the previous exercises to:

1. Select a scenario from the ALE Gymnasium library. You are free to choose any available scenario, but for this part of the project, a **more complex environment** than the one chosen in the first part is required. Keep in mind that the complexity of the selected environment will be considered when grading this activity.

    - Gymnasium (https://gymnasium.farama.org/)
    - ALE (https://ale.farama.org/)

2. Select two or more models and compare their performance in the chosen environment. You may use available implementations from **libraries** or **third-party code** for this task.

    For instance, you can have a look at:

    - Stable Baselines 3[8] (SB3) [Raffin et al., 2021] is a popular library providing a collection of state-of-the-art RL algorithms implemented in PyTorch. It builds upon the functionality of OpenAI Baselines (Dhariwal et al., 2017), aiming to deliver reliable and scalable implementations of algorithms like PPO, DQN, and SAC.
    - CleanRL[9] [Huang et al., 2022] is designed to provide clean, minimalistic implementations of RL algorithms. It focuses on simplicity and transparency, making it easier for researchers to understand and experiment with different RL techniques.

[8] https://github.com/DLR-RM/stable-baselines3
[9] https://github.com/vwxyzjn/cleanrl

- Tianshou[10] [Weng et al., 2022] is a versatile library for RL research that supports various training paradigms, including off-policy, on-policy, and multi-agent settings. It offers a modular design that allows users to easily customize and extend the library's components.
- Ray Rllib[11] [Liang et al., 2018] is part of the Ray ecosystem and is known for its scalability and support for distributed RL training. Rllib provides a diverse range of algorithms and tools for both single-agent and multi-agent scenarios.
- Dopamine[12] [Castro et al., 2018] is a research framework developed by Google for experimenting with reinforcement learning algorithms. It is designed to provide a clean, minimalistic codebase that focuses on implementing and evaluating RL algorithms such as DQN and its variants. Dopamine emphasizes reproducibility and simplicity, offering well-structured, modular components that make it easy for researchers to implement and test new algorithms.

Further information could be found at [Kwiatkowski et al., 2024].

Questions:

1. Describe the **selected environment** used in the experiment, outlining its main characteristics and objectives.
2. Describe the **preprocessing steps and wrappers** applied in your experiment, specifying the resulting observation space.
3. Provide a **description** of the chosen agents or models, including their architecture, learning algorithms, and relevant design decisions.
4. **Train** the models, applying fine-tuning or **hyperparameter optimization** when necessary to improve performance.

[10] `https://github.com/thu-ml/tianshou/`
[11] `https://docs.ray.io/en/latest/rllib/index.html`
[12] `https://github.com/google/dopamine`

5. **Save** the best-performing models and produce **visualizations** that illustrate their learning curves and in-environment behavior.

6. **Evaluate** the trained models within the environment, reporting their **average success rate** and any relevant performance metrics.

7. **Present** the results, **analyze** the findings, and **justify** the selection of the model that achieves the best performance for this environment.

5 Delivery format

Students MUST deliver the following documentation (compressed in a ZIP file):

1. A **PDF document** (including the names and NIUs of all group members) containing detailed responses to all the questions listed above.

   The document should incorporate images, graphs, plots, code snippets, and any other relevant materials necessary to explain and justify your answers clearly.

2. A set of **slides** (in PowerPoint or PDF format) prepared for your class presentation.

   The slides should include images, videos, and other supporting materials that effectively highlight your project, methodology, and results.

3. A folder containing the **source code**, or alternatively, a link to a version-controlled repository (e.g., GitHub[13] or an equivalent platform).

   The submission may include Jupyter Notebooks (.ipynb) or Python scripts (.py), but it **must** satisfy the following requirements:

   - The code must be well-commented, with clear explanations that facilitate readability and understanding.
   - A file named "requirements.txt" must be included, listing all the necessary libraries and their corresponding versions to ensure correct execution.
   - A "README.md" file must be provided with detailed instructions on how to run each part of the project (e.g., training, testing). Any component that cannot be executed following these instructions will receive a grade of 0.

[13] `https://github.com/`

6 Presentation

Each group is required to deliver a **presentation** of their project to the rest of the class.

Presentation Criteria:

1. Each presentation will have a duration of **10 minutes**, followed by **10 minutes** for questions and discussion with the class.

2. You must use **appropriate supporting materials**—such as slides, code demonstrations, videos, or visualizations—to effectively present your work and highlight the results obtained.

3. Presentations will be held on December 17 and December 20, during the regular class schedule.

References

[Castro et al., 2018] Castro, P. S., Moitra, S., Gelada, C., Kumar, S., and Bellemare, M. G. (2018). Dopamine: A research framework for deep reinforcement learning.

[Huang et al., 2022] Huang, S., Dossa, R. F. J., Ye, C., Braga, J., Chakraborty, D., Mehta, K., and Araújo, J. G. (2022). Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18.

[Kwiatkowski et al., 2024] Kwiatkowski, A., Towers, M., Terry, J., Balis, J. U., Cola, G. D., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Tan, H., and Younis, O. G. (2024). Gymnasium: A standard interface for reinforcement learning environments.

[Liang et al., 2018] Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Goldberg, K., Gonzalez, J., Jordan, M., and Stoica, I. (2018). RLlib: Abstractions for distributed reinforcement learning. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3053–3062. PMLR.

[Raffin et al., 2021] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8.

[Weng et al., 2022] Weng, J., Chen, H., Yan, D., You, K., Duburcq, A., Zhang, M., Su, Y., Su, H., and Zhu, J. (2022). Tianshou: A highly modularized deep reinforcement learning library. *Journal of Machine Learning Research*, 23(267):1–6.