COMP6008 Reinforcement Learning
Group Project, Presentation, and Report Guidelines

August 7, 2025

1 Important

- This version of the assignment specification may be replaced, in whole or in part, by updates posted to `Blackboard`, either as new versions, as updates on the assignment page and/or as announcements posted to the `Blackboard` Announcements page.
  - The version date appears at the title of assignment specification (this document) and on the `Blackboard` Assessments/Project page.
  - Please make sure you follow the latest assignment specification as it may affect what you are expected to turn in.
  - No accommodation will be made for marks lost because you followed an outdated assignment specification.
- The project report will be checked through `Turnitin`.
  - The similarity with existing materials should be less than **15%** (excluding references and cover page), otherwise it will be treated as plagiarism and zero mark will be received for your project.
  - Please note NO similarity report will be available for the students after submission.
  - Generative AI tools such as ChatGPT, Gemini, Claude, etc., cannot be used for your report writing. Should a high percentage of AI-generated text be detected, an interview will be scheduled to assess both your authorship and comprehension of the report. Failure to demonstrate a sufficient understanding of the content will result in the failure of the project.
- **Timeline**:
  - Project Implementation and Presentation: 23:59, October 17, 2025 (Friday, Week 13).
  - Project Report (`Turnitin` submission): 23:59, October 24, 2025 (Friday, Week 14).

2 Objectives

In this project, students will work in groups to apply Reinforcement Learning (RL) technique(s) to solve specific problem(s), critically analyse the results, and present both the findings and the analysis. Additionally, students need to propose potential improvements or expansions to the existing work and provide justification for their proposals. The project aims to achieve the following learning outcomes:

- Gain a solid understanding of core concepts in reinforcement learning, such as Markov Decision Process, reward functions, value functions, policy, exploration-exploitation trade-offs.
- Learn how to model complex decision-making problems as Markov Decision Processes (MDPs), including defining the state space, action spaces, and designing reward function, and implement them in code.
- Develop hands-on skills in implementing (or adapting) and fine tuning reinforcement learning algorithms covered in the unit, and understand the mathematical foundations behind these algorithms.
- Acquire the experience of selecting appropriate reinforcement agents to solve complex decision making problems, and evaluating their performance through metrics such as average return(or reward), convergence speed, and sample efficiency, generalisation to unseen environments.

- Develop proficiency in identifying issues that arise during the implementation and training phases, and resolving these issues or proposing potential solutions with sound justification.
- Practice documenting project progress, including code documentation, experimental results, and insights gained, and effectively communicate findings through presentations or reports to both technical and non-technical audiences.
- Improve skills in group project planning, time estimation, task prioritisation, and resource allocation to ensure the timely completion of project milestones and deliverables.

3 Summary of Project Requirements

1. Form a group of 3 students for your project study.
   - Only in exceptional cases, different group size is allowed but it requires permission from unit coordinator.
   - If you have not found a group, you can post a thread under `Blackboard` Discussion Board to express that you are recruiting teammates or available to join a group.
2. The group choose one of following two options for the project:
   - Option A: Focus on the problem.
     - Select a real-world problem (can be from employer), and get the approval from unit coordinator.
     - Reformulate the problem using RL framework.
     - Select and implement suitable RL agent to solve it.
     - Analyse the performance.
   - Option B: Focus on the RL.
     - Select at least three well-documented problems.
     - Implement at least three RL agents to solve **each of them**.
     - Evaluate and compare their performance.
3. Submit the well-commented project code along with documented instructions on how to run it. The submission worth 30 Marks and is due by 23:59, October 17, 2025 (Friday, Week 13)).
4. Record a presentation of the project findings, with a maximum length of 20 minutes. The presentation worth 10 Marks and is due by 23:59, October 17, 2025 (Friday, Week 13)).
5. Write a group report that includes (due by 23:59, October 24, 2025 (Friday, Week 14))
   - Description and critical review of the project (worth 20 Marks).
   - Potential improvements AND expansion of existing work, with scientific justification (worth 15 Marks).
   - Statements of individual members' contributions to the project are required. While no marks are allocated for this statement, failure to provide it will result in penalties for the group.
6. Finally, each group member submits a peer review report. This submission is optional if the group agrees that all group members contributed equally; otherwise, it is compulsory.

4 Project Selection

4.1 Option A: Focus on the Problem

This option is for students who want to solve real-world decision making problems using RL techniques. Students are expected to

- Select a real-world problem that is not trivial (that is, the solution is not obvious). The group is highly recommended to run the selected problem past the unit coordinator (UC) before going too much further into the project study to make sure that it isn't too simple or complicated. Otherwise, the group will lose marks and even fail the unit if the UC deems the selected topic as inappropriate.

- Formulate the selected problem as a RL environment. This formulation requires careful analysis and you can go through following questions before you start implementing the environment:
  - Does the environment emit observations?
    - If yes, what are the observation properties (continuous/discrete, dimensionality, range)?
    - If no, is it a multi-armed bandit problem?
  - Does the environment emit rewards?
    - If yes, are they suitable for RL (must be a number, not too sparse)?
    - If not, are you able to design suitable rewards that would allow RL agent to solve the environment?
  - Does the environment accept actions that affect its evolution?
    - If yes, what are the action properties (continuous/discrete, dimensionality, range)?
    - If no, it is not RL problem
  - How much time does it take for the environment to process one time step (receive action and emit observation and reward)? Remember that it may take RL agent millions of time steps to solve an environment.
  - Can you implement the environment in a reasonable time (avoid special algorithms/obscure data)?
- Implement the environment, i.e., the probability $p(s_{t+1}, r_{t+1}|s_t, a_t)$, in a Python notebook with at least following details
  - Observation space definition. Please note that the observation space defined here differs from the information observed by an agent at each time step. For instance, a robot's sensor may only output positional information at time step t as z_t. However, you can concatenate measurements from multiple time steps to define the observation space, such as $o_t = \{z_t, z_{t-1}, z_{t-2}\}$.
  - Reward signal emitted from environment.
  - Actions that an agent can take to impact the environment.
  - Transition model of the environment to the next state.

  It is recommended to use the gymnasium.Env class.
- Select a suitable agent for the selected environment. The implementation of the agent can utilize existing RL libraries, such as Stable Baselines3 (link), Ray RLlib (link), Tianshou (link). Both Ray RLlib and Tianshu have off-line RL algorithms that can be used to problems only offline data is available, for example, stock market data. Alternatively, you may opt to use our practical implementations if they are suitable for solving the selected problem
- Perform multiple runs to solve the environment, including hyperparameter fine-tuning. It's possible that the RL agent may fail to solve the problem due to various factors beyond your control. However, it's crucial to note that your project can still achieve a high score if you provide a scientific justification for the failure and suggest potential improvements to the selected agent. Simply attributing failure to limited computational resources before the deadline is not sufficient.

4.2 Option B: Focus on the RL

This option is for students who want to evaluate performance of various agents in a number of well-documented RL environments, and you are expected to

- Select at least three different RL environments that are not trivial. You are recommended to use existing implementations of known environments using OpenAI Gymnasium (e.g, robotics in MuJoCo or simple Atari games). DO NOT choose environments in Toy Text and Classic Control groups, and Lunar Lander environment that we have used in Practicals.
- Choose three different RL agents covered in this unit for each environment and implement them in Python. You may choose the same three agents and apply them across all environments if deemed appropriate. You must use our implementations from Practicals as the starting point and include your own modifications/ideas in the agent code. You CANNOT use available implementations of agents other than than those in our Practicals. For example, you can implement the SAC from using the REINFORCE agent in the practical as the starting point.

- Perform multiple runs to solve the environments by all the agents. You are expected to tackle these well-known RL problems, considering the wealth of publicly available solutions, or at the very least, demonstrate the potential of solving them using selected RL agents. However, even if you encounter difficulties in solving the problems, you can still attain high marks by providing a scientific justification for the failure. It's important to note that reasons for failure cannot simply be attributed to a lack of computational resources.

You need to make sure that the chosen environments can be interacted with and solved (at least in principle) by your agents. Before you start working on the implementation, make sure that:

- Your agents can receive observations and take actions as required by the environments. For example, robotics environments typically require continuous actions, but many RL methods can only select actions from a discrete set.
- Your agents can process observations and take actions in a reasonably short amount of time. Remember that it may take RL agent millions of time steps to solve an environment, especially if observations are high-dimensional (e.g., images).

4.3 Examples

4.3.1 Example 1 - Stock Trading

This is an example of a project under option A. The goal is to maximise returns while managing risk when trading stocks. The RL agent need to allocate capital across multiple stocks. To complete this project, you would have to

- Collect data about stock market (e.g., Australia Stocks Dataset)
- Use the data to simulate environment (users).
  - Note 1: what is state space (context) definition
  - Note 2: actions are continuous and constrained by your maximum budget.
- In the data, actions have no effect towards the world, hence, off-policy and offline agents are expected to achieve good performance in testing data.

4.3.2 Example 2 - Active Suspension Control

This is an example of a project under option A. The goal is to learn actuator forces applied at the wheel assemblies of a car in order to maximise handling or passenger comfort. The RL agent is a controller that sets the desired forces (action). The environment includes car dynamics and road pattern, and emits sensory data to the agent such as car body acceleration and suspension deflection (observation). To complete this project, you would have to:

- Implement quarter-, half- or full-car active suspension model that takes as input current suspension state and actuator forces, and returns the suspension state at the next time instant.
- Design a reward that represents the learning goal (e.g., square of car body acceleration if the goal is to maximise passenger comfort)
- Note 1: state space is continuous (low-dimensional).
- Note 2: actions can be made continuous or discrete.

This type of project requires you build a simulation platform, unless you have access to a real platform.

4.3.3 Example 3 - Robotics

This is an example of a project under option B. The goal is to evaluate various agents in robotics control implemented in the OpenAI Gymnasium MuJoCo group (link), for example:

- Inverted double pendulum (link).
- Swimmer (link).
- Half-cheetah (link).

All these environments are characterised by action and state spaces that are multi-dimensional and continuous. This means that either you must use agents that are capable of taking continuous actions or you must discretise the action space and implement agents that take discrete actions. For example, you could:

- Implement Soft-Actor-Critic agent suitable to continuous action spaces.
- Discretise the action space and implement DQN and basic policy gradient agents for discrete action spaces. This approach is generally not recommended due to its lower performance, but it is acceptable for students who prefer to avoid significant code changes in the practicals.

5 Assignment

This assignment comprises two components: the project code submission in 5.1 and the presentation in 5.2.

5.1 Project Code (30 Marks)

The project code should be submitted before 23:59, October 17, 2025 (Friday, Week 13) and

- Prioritises readability and clarity.
  - Organize your code into a single, well-structured notebook.
    - Sections for each major component, such as environment setup, agent implementation, training loop, and result analysis.
    - Explanations for the logic flow between sections.
  - Organise the notebook with clear sections, the logic flow between sections should be smooth.
  - Use markdown cells to provide context, explanations, and instructions.
  - Code cells are accompanied by clear explanation and comments.
  - Present your results clearly using plots, graphs, or tables. Visualisations should be well-labelled and easy to interpret. Examples include learning curves (average reward over time), convergence plots, or tables comparing agent performance.
  - Demonstrate the process of hyperparameter fine-tuning and its impact on the agent's performance.
- Contains clear instructions for setting up the environment and reproducing results, if you are not using `Google Colab`.
  - include a "requirements.txt" file for your project (in the console type: pip freeze > requirements.txt).
  - include instructions to install dependencies and obtain additional data if they are required.
- Provides clear and well-structured pseudo-code for the selected RL agents (and a mathematical description of the environment dynamics for Option A).
- Implements the chosen RL algorithms/methods comprehensively and correctly, ensuring that the code aligns with the provided pseudo-code (and with the mathematical formulation of the environment process for Option A).

If students choose Option B and **make only minor modifications to the provided practical code** to solve the selected problems, **In this case, the submitted project code will not be formally assessed. Instead, the final report will carry greater weight and be marked out of 65, replacing the usual breakdown of 30 marks for code and 35 marks for the report**. Minor modifications includes

- Changes to the hyperparameters of the selected agent, such as the learning rate or the input/output dimensions of the neural network.
- Modifications to the implementation of existing functions in the provided practical code (without adding new functions), for example altering the replay buffer structure in the DQN based on the PyTorch DQN tutorial.
- Modifications to the generation of plots or graphs.

Although the submitted code will not be assessed in this case, students are still required to submit the project code to Blackboard. Marking rubric is provided in Table 1, and students are required to read it carefully.

Table 1: Project Code Marking Rubrics

Criteria	Excellent ($\geq$85%)	Good (70-85%)	Satisfactory (50-69%)	Unsatisfactory (<50%)
Notebook organisation and structure (5%)	Notebook is well-organised with clear sections, logical flow between sections.	Notebook is organised with distinct sections but may lack some cohesion between them.	Sections are present but not well-structured. Flow between sections is unclear at times.	Notebook lacks organisation; sections are unclear or missing.
Clear Explanation of Code (10%)	Code cells are accompanied by clear explanations and comments, also markdown cells are provided to aid understanding of the purpose and functionality of the code.	Code explanations are present but may lack clarity or detail in some parts.	Code explanations are brief or somewhat unclear. Some code cells lack context.	Limited code explanations; code cells are often lacking context.
Implementation of Reinforcement Learning (40%)	Comprehensive and correct implementation of chosen RL algorithms/methods (and chosen environment modelling for Option A), and match between pseudo code and implemented algorithm (or environment model)	Minor errors may be present, but they don't significantly affect results.	Basic implementation with notable errors or omissions.	Implementation is incomplete, leading to errors in results.
Experimentation and Result Presentation (35%)	Thorough experimentation with comprehensive results. Clear presentation using plots, graphs, or tables.	Reasonable experimentation with clear presentation of results. Some minor gaps in the presentation.	Limited experimentation with unclear presentation of results.	Minimal experimentation and unclear presentation.
Reproducibility and Environment Setup (10%)	Clear instructions for setting up the environment and reproducing results. Dependencies are clearly listed.	Instructions for environment setup and result reproduction are present, but some details may be lacking. Dependencies are mostly listed.	Setup instructions are somewhat unclear. Dependencies may be missing or incomplete.	Setup instructions are unclear or inaccurate. Dependencies are not properly listed. The student will receive **ZERO** mark for project, and therefore **fails** the unit.

Note 1: If `Google Colab` is used, the criterion *Reproducibility and Environment Setup* will not be assessed. Instead, marks for the remaining criteria will be scaled up to a total of 30. However, a zero mark penalty will still apply if the results presented in the presentation and report are not included in the notebook.

Note 2: For Option B, if fewer than nine agent-environment combinations are implemented, a penalty will be applied. A total of nine applications—i.e., three agents applied across three distinct environments—is expected. The final code mark will be adjusted using the following formula:

$$\text{adjusted Code Mark} = \left(\frac{\text{Number of Combinations}}{9}\right) \times \text{Code Mark Based on Rubric}$$

5.2 Presentation (10 Marks)

Each group is required to submit a recorded presentation before 23:59, October 17, 2025 (Friday, Week 13). It is recommended to use PowerPoint and instructions are available here. The presentation should strictly adhere to a maximum duration of 15-20 minutes per group (penalties will be applied for exceeding this time limit) and include ONLY:

- Option A
  - A brief description of the problem to be solved, and explain
    - why the problem is important?
    - why it is challenging/difficult?
    - why RL is suitable to solve the problem?
  - Give a detailed explanation of your design and implementation of the environment that represents the problem, it should include
    - environment state definition.
    - reward design and why it match the final goal to be achieved by RL agents.
    - action space definition.
    - mathematical description of the environment dynamic model $p(s'|s,a)$ OR data source used to simulate the model $p(s'|s,a)$.

    Please note that the presentation should also include the linkage between code snippets and the theories that guide your implementation.
  - A brief description of the RL agents that you plan to deploy, and explain why the agents are selected, i.e., why the agent can solve the problem.
  - Present training results with learning curves (average reward over time), convergence plots, or tables comparing agent performance.
  - Marking rubric for this option are available in Table 2.
- Option B
  - A brief description of the problems to be solved, including the definitions of state and action spaces, reward signals, and goal to be achieved for the problems.
  - Detailed description of the selected agents, at least including
    - theories behind the agents' implementation.
    - linkage between code snippets and the matched theories.
    - highlight of the modifications made to the original codes provided in the Practicals.
    - justification that the selected agents can solve the chosen problems.
  - Present training results with learning curves (average reward over time), convergence plots, or tables comparing agent performance.
  - Marking rubric for this option are available in Table 3.

Table 3: Presentation Marking Rubrics (Option B)

Criteria	Excellent (≥85%)	Good (70-85%)	Satisfactory (50-69%)	Unsatisfactory (<50%)
Problem Description (15%)	Thoroughly and concisely explains the problem, including correct definitions of reward, state, action, and original goal, demonstrating deep understanding and engaging the audience.	Clearly explains the problems with definitions of reward, state, action, and original goal, showing a good understanding.	Describes the problem in a basic manner, with noticeable gaps or errors in defining essential components of the problems.	Provides a vague or unclear description of the problem. Less than three problems were introduced.
RL Agent Modifications/Implementations (45%)	Presents detailed modifications/implementations of RL agents with theoretical foundations explained, and clearly linking them to the problem's requirements.	Presents well-explained modifications/implementations of RL agents, effectively addressing the problem, but lack correct description of the theories	Presents modifications/implementations, but with some minor gaps in connection to the problem.	Presents modifications/implementations with noticeable gaps or lack of clear relevance to the problem.
Results presentation (20%)	Training results are presented with clear, well-labeled learning curves, convergence plots, and/or comparative tables. Visuals are accurately interpreted, and key insights are discussed (e.g., convergence speed, stability, agent performance). Results support conclusions in the report.	Includes useful plots or tables showing training progress and agent performance. Visuals are mostly clear and labelled, with basic interpretation. Some insights are mentioned, though limited in depth.	Includes training results with basic learning curves or tables, but may lack clarity, labels, or sufficient explanation. Interpretation is minimal or superficial.	Includes incomplete or unclear plots/tables with little to no interpretation. Visuals may be misleading or not relevant to performance evaluation.
Clarity(20%)	Speaks clearly and articulately throughout the presentation, ensuring all content is easily understood.	Generally speaks clearly, with only minor instances of unclear speech.	Speech is mostly clear, but with some instances of unclear delivery.	Consistently unclear speech, significantly impeding understanding of the presentation.

6 Project Report (35 Marks)

The project report should be split into three parts: (1) a description and critical review of the project, (2) proposals for potential improvements and expansion of the existing work, and (3) a statement of individual contributions, and it should be submitted before 23:59, October 24, 2025 (Friday, Week 14).

6.1 Part 1: Description and Critical Review (20 Marks)

This part should be based what you have achieved throughout your project, and you need to

- Option A:
  - Provide a background introduction to the selected problem, explain why it is important for the project to be solved, and what are the challenges to solve the problem using RL techniques. The importance of the project can be highlighted by outlining the benefits to society, or the potential contribution to expanding knowledge in the field of RL.
  - Give a detailed description of the Markov decision model for the selected problem, include

    * The state and action spaces design with scientific justification.
    * Reward signal design and explain why shaped reward signal will help to achieve the final goal of the selected problem and speed up the training of RL agents.
    * State transition model from current state to next state. It can be mathematical model from physics law, simulation software, or data driven.
  - Justify of selected agent, and describe of the process of tuning hyper-parameters.
  - Analyse the training results, and validate the agent in the testing environment or testing data.
  - Critically review the projects work. For examples, has the agent solved the problem? If yes, which part of your work contribute to the success. If not, what are the potential causes to the failure? What are the implication of achieved results?
  - The marking rubric is available in Table 4.

- Option B:
  - Describe the objectives of the project. For example, compare the performance of DQN, SAC, PPO on environments with discrete action spaces, considering factors such as data efficiency, stability of convergence, generalisation, etc. Also, highlight the importance of the project by outlining its benefits to society or its potential contribution to expanding knowledge in the field of RL.
  - Briefly describe essential information about the selected environments, including all variables in the Markov decision model and their properties, and explain why these environments are suitable for the purposes of your project study.
  - Explain the theories behind selected RL agents, and explain why these agents can solve the problem. Also illustrate the process of training these agents, and tuning hyper-parameters.
  - Analyse the training results, and validate the agent in the testing environment or testing data.
  - Critically review the project's work. For example, has the agent solved the problem? If so, which part of project work contributed to the success? How do the agents and their hyper-parameters affect the performance? If no success was achieved, what are the potential causes of the failure? What are the implications of the achieved results in terms of meeting the study objectives?
  - Thex marking rubrics are available in Table 5.

Table 5: Marking Rubrics of Project Report - Part 1 (Option B)

Criteria	Excellent (≥85%)	Good (70-85%)	Satisfactory (50-69%)	Unsatisfactory (<50%)
Objectives Descriptions (10%)	Comprehensive description of the study objectives, including its significance and relevance. Clearly articulates the challenges to be overcome.	Study objective description is presented but lacks depth or clarity. Importance may not be fully conveyed.	Study objective description is somewhat unclear. Importance and challenges are vaguely stated.	Study objective description is missing or unclear. No context, importance, or challenges presented.
Environment Model (10%)	Detailed explanation of the environment model that relevant attributes, states, actions and rewards, are correctly defined with scientific justification.	Clear environment description with relevant attributes, states, actions, and rewards. Some minor details might be missing.	Environment description is present but lacks clarity or completeness for some parts of the problem definition.	Environment description is vague or lacks important details.
Agent Selection and Training (25%)	Comprehensive discussion of the process of choosing agents, covers accurate theories behind the agents, and the process of tuning agent parameters. Clear reasoning behind parameter choices and their impact on performance.	Clear explanation of choosing agents and parameter fine-tuning process, though some aspects could be more detailed. Some reasoning behind choices is provided and lack theoretical explanation of agents.	Processes of agent selection and parameter fine-tuning are outlined, but lacks depth or clarity. Limited reasoning for parameter choices and discussion of theories.	Parameter fine-tuning process and agent selection are somewhat unclear. Little reasoning for choices provided and no theories is covered.
Results and Outcome (25%)	Thorough presentation of results, including quantitative and qualitative outcomes. Clear insights drawn from the results and their relevance to the problem and chosen agents.	Clear presentation of results with insights drawn, though some analysis might be lacking depth. Relevance to the problem and chosen agents is evident.	Presentation of results is present, but some details may be unclear. Insights and analysis may lack depth.	Results are presented, but their relevance to the problem and chosen agents is not fully explained. Insights may be superficial.
Critical Review of Work (20%)	Thorough and thoughtful critical review of the project's work, including the match between expected and achieved results. Potential implications are deeply considered and discussed.	Comprehensive review of work, discussing alignment between expectations and outcomes, but depth may vary.	Basic review of work with some discussion on results and possible implications. Analysis lacking depth.	Limited critical review; discussion on results and implications is superficial.
Presentation and Clarity (10%)	Well-organized and visually appealing presentation of the problem, environment, agent(s), training, results, and review. Clear explanations with minimal errors. Use the IEEE format without errors.	Neat presentation with organized content. Explanations are generally clear, though some sections might lack clarity. Minor errors in writing. Minimal inconsistency in IEEE format	Presentation is somewhat disorganised, affecting overall clarity. Explanations may be unclear in parts. Noticeable errors in writing. Major inconsistency with IEEE format.	Presentation is messy or inconsistent. Explanations lack clarity and are hard to follow. Significant errors in writing. Non IEEE format is used.