Reinforcement Learning Assignment

COMS4061A/COMS7071A

Sergio Frasco (2427724@students.wits.ac.za)
Muhammad Goolam(2454262@students.wits.ac.za)
Rayhaan Hanslod(2430979@students.wits.ac.za

Due date: 22 October 2025, 23:59

1 Introduction

Open-world survival games present complex challenges for reinforcement learning agents, requiring them to balance multiple objectives such as resource gathering, crafting, exploration, and survival. These environments test an agent's ability to learn hierarchical behaviours, long-term planning, and adaptive strategies in procedurally generated worlds with sparse rewards.

Your task will be to implement an agent that can survive and thrive in the Crafter environment, learning to manage hunger and health while progressively unlocking achievements through exploration, resource collection, crafting, and combat. The agent must learn to prioritise different objectives based on its current state and navigate the trade-offs between immediate survival needs and long-term progression goals.

2 Environment

Crafter is a procedurally generated 2D survival game designed as a benchmark for reinforcement learning research. It features a diverse set of tasks including resource gathering, tool crafting, creature combat, and achievement hunting, all while managing survival mechanics like hunger and health. The environment provides both dense survival rewards and sparse achievement rewards, making it an ideal testbed for evaluating generalisation and exploration in RL agents.

The GitHub repository can be found at https://github.com/danijar/crafter and the research paper detailing the environment can be found at https://arxiv.org/pdf/2109.06780. Look at these for installation and setup instructions (Crafter is compatible with the Gymnasium interface, which you are required to use in this assignment). The paper provides detailed information about:

- The observation space (64x64 RGB images)
- The action space (17 discrete actions)

- The reward structure and 22 achievements
- Survival mechanics and game dynamics

You can find the skeleton code here:
https://github.com/rayrsys/Reinforcement-Learning-Project-2026-Crafter.git

3 Requirements

You are allowed to work in groups of at most 4 people. Each group will be required to do the following for this assignment:

1. Implement one learning algorithm from the course, and one algorithm that has not been covered in the course (if in doubt, ask one of the tutors). These **two** agents will serve as a base from which you will make improvements.
2. For the algorithm that has not been covered in the course, you need to explain the algorithm in your report, to show that you understand how it works.
3. The **core** part of the assignment will be to iteratively improve the base agents. Specifically, you will need to implement the base design, evaluate its performance, identify areas for improvement, and then make those improvements. This iterative process will need to be done a few times (i.e. simply making one improvement will not be sufficient).
   (a) For each iteration, you will need to present the results and analysis of your agent's performance, and use this to motivate the design changes to make for the next iteration.
   (b) The improvements cannot be trivial. They need to be some change to either the learning algorithm, the policy model, some preprocessing of the various elements of the MDP (e.g. action or observation spaces, the reward function) etc. *For example, one such improvement may be to use image preprocessing or feature extraction on the observations. However, simply changing hyperparameters is insufficient.*
   (c) We require a minimum of **TWO** improvements to be made — you are allowed to make further improvements, and these will be considered for extra credit. So, at a minimum, your investigative pipeline would be:

	implement base agent	
$\rightarrow$	evaluate agent	**(Eval 1)**
$\rightarrow$	make a potential improvement	**(Improvement 1)**
$\rightarrow$	evaluate agent	**(Eval 2)**
$\rightarrow$	make a potential improvement	**(Improvement 2)**
$\rightarrow$	evaluate agent	**(Eval 3)**

   (d) **Remember that this iterative process of improvement needs to be done for both of your chosen algorithms.**

4. Once you have completed all your improvements to each agent, compare their final performance and their strengths and weaknesses. This could include points like how well each agent achieves the goal, and their rates of learning.

5. If you use any existing research or code in any of your solutions, you are required to cite it, AND make some kind of modification to it (e.g. shaping the reward, augmenting input, using sub-task curricula etc.) and describe it in the report.

We are mainly interested in your approach to the solution, not necessarily in how well you can solve the problem. So make sure your agent evaluations are complete and provide evidence for your design decisions, which will then be reasonable.

When you are presenting results, please choose a suitable format to convey them. For example, don't present the return over episodes in a table; rather use a line graph.

4 Particulars and Restrictions

Please take note of the following:

1. You are allowed to use libraries (like Stable Baselines3) to implement and train your agents.

2. A YAML file with the correct package versions has been provided. Please use these versions to ensure compatibility.

3. When installing and setting up Crafter, please ensure you use a Gymnasium environment interface. Using a Gymnasium environment will allow you to interface with libraries more easily.

4. Use the standard Crafter environment with partial observability (e.g. when calling `gym.make("CrafterPartial-v1"`

5. The main rewards you will need to optimise consist of:

   (a) Survival reward: The agent receives +1 reward for each timestep it survives.

   (b) Achievement rewards: One-time rewards for unlocking each of the 22 achievements (e.g., collecting wood, crafting tools, defeating enemies).

   (c) The standard Crafter environment provides these rewards automatically, and agents should learn to balance immediate survival with achievement progression.

   Please note that you are allowed to alter this reward (e.g. through reward shaping), but for all evaluation you do, please include performance on the standard rewards and achievement unlock rates.

6. We have provided a wrapper to create the Gymnasium environment that converts between different Gym API versions for compatibility with Stable Baselines. It has been well-commented with what you can and can't change, and what you should be completing yourself.

(a) The wrapper is minimal and primarily handles API compatibility between Gymnasium and older Gym versions.

(b) If you want to preprocess observations (e.g., feature extraction from images) or modify the action space, you will need to implement this yourself.

(c) Please note that the default observation space is 64x64 RGB images. You may want to investigate different preprocessing techniques or feature extraction methods to improve learning efficiency.

(d) Depending on the 3rd party library you are using, you may need to adapt your observations and actions to make them compatible.

7. The Crafter environment has built-in visualisation tools and logging capabilities. You can use these to analyse your agent's behaviour and track achievement progress throughout training.

5 Evaluation

Marking will be done on the process of making your design decisions based on prior results, and not necessarily on how well your agent is able to perform.

However, you also need to submit your source code, so your results need to be reproducible if we test your agents ourselves.

We will be evaluating your agents' performance using the following metrics from Crafter:

- Achievement unlock rate: The percentage of times each achievement is unlocked across episodes
- Geometric mean of achievement unlock rates: The overall score combining all achievements
- Survival time: Average number of timesteps survived per episode
- Cumulative reward: Total reward accumulated per episode

Please make sure to include results using these standard metrics throughout your investigation.

6 Submission

Each group will be required to submit the following:

1. A report, detailing the content specified in Requirements. Only one group member should submit, but all group members' names and student numbers must be specified at the top of the report.
2. The source code. Marks will be awarded for the quality of your codebase — things like comments, Read Me files (with an explanation of your directory structure), run scripts, and code versioning. You should include a link to the GitHub repository in your report.

3. Make sure to include the hyperparameters used for your training in your report.