Durham University

Computer Science

Summative Assignment

Module code and title	COMP3667 Reinforcement Learning
Academic year	2025-26
Coursework title	Setup
Coursework credits	1 credits
% of module's final mark	10%
Lecturer	Robert Lieck
Submission date*	Thursday, 19 February, 2026 14:00
Estimated hours of work	2
Submission method	Gradescope

Additional coursework files	• *Reinforcement Learning Paper Template (Overleaf)* • *Coursework Template Starter Code (IPython Notebook)*
Required submission items and formats	*xxxx00.zip* • *xxxx00-agent-paper.pdf* • *xxxx00-agent-code.ipynb (or .py)* • *xxxx00-agent-code-hardcore.ipynb (or .py)* • *xxxx00-agent-log.txt* • *xxxx00-agent-log-hardcore.txt* • *xxxx00-agent-video,episode=210,score=85.mp4* • *xxxx00-agent-video-hardcore,episode=350,score=-92.mp4*

* This is the deadline for all submissions except where an approved extension is in place. For assessments carried out in weekly practicals or other scheduled sessions, the date shown will be the Monday of the week in which the assessments take place.

Late submissions received within 5 working days of the deadline will be capped at 40%.
Late submissions received later than 5 days after the deadline will receive a mark of 0.
It is your responsibility to check that your submission has uploaded successfully and obtain a submission receipt.

Your work must be done by yourself (or your group, if there is an assigned groupwork component) and comply with the university rules about plagiarism and collusion. Students suspected of plagiarism, either of published or unpublished sources, including the work of other students, or of collusion will be dealt with according to University guidelines (https://www.dur.ac.uk/learningandteaching.handbook/6/2/4/).

Durham University

Computer Science

Summative Assignment

Module code and title	COMP3667 Reinforcement Learning
Academic year	2025-26
Coursework title	Final Submission
Coursework credits	9 credits
% of module's final mark	90%
Lecturer	Robert Lieck
Submission date*	Tuesday, 28 April, 2026 14:00
Estimated hours of work	18
Submission method	Gradescope

Additional coursework files	• *Reinforcement Learning Paper Template (Overleaf)* • *Coursework Template Starter Code (IPython Notebook)*
Required submission items and formats	*xxxx00.zip* • *xxxx00-agent-paper.pdf* • *xxxx00-agent-code.ipynb (or .py)* • *xxxx00-agent-code-hardcore.ipynb (or .py)* • *xxxx00-agent-log.txt* • *xxxx00-agent-log-hardcore.txt* • *xxxx00-agent-video,episode=210,score=85.mp4* • *xxxx00-agent-video-hardcore,episode=350,score=-92.mp4*

* This is the deadline for all submissions except where an approved extension is in place. For assessments carried out in weekly practicals or other scheduled sessions, the date shown will be the Monday of the week in which the assessments take place.

Late submissions received within 5 working days of the deadline will be capped at 40%.
Late submissions received later than 5 days after the deadline will receive a mark of 0.
It is your responsibility to check that your submission has uploaded successfully and obtain a submission receipt.

Your work must be done by yourself (or your group, if there is an assigned groupwork component) and comply with the university rules about plagiarism and collusion. Students suspected of plagiarism, either of published or unpublished sources, including the work of other students, or of collusion will be dealt with according to University guidelines (https://www.dur.ac.uk/learningandteaching.handbook/6/2/4/).

Learning to Walk Efficiently

Reinforcement Learning Summative Assignment

Epiphany Term 2026

Overview

Lecturer	Robert Lieck	
Coursework component	Setup	Final Submission
Coursework credits	1 credit	9 credits
% of module's final mark	10%	90%
Estimated hours of work	2 hours	18 hours
Submission method	Gradescope	

Introduction

This assignment consists of two parts, the *Setup* (10%) and the *Final Submission* (90%). Both are identical in terms of what has to be submitted (number and structure of files, see below) and differ only in how it is being evaluated. For the *Setup*, completeness and formal correctness of the submission is the main criterion, the actual performance secondary. The *Final Submission* also has to be complete and formally correct, but the evaluation focuses on the quality of the results and the report.

The assignment is to train a bipedal robot to learn to walk efficiently in a 2D physics simulation. It should be *sample efficient*, learning with the least amount of interaction with the environment as possible, and also it should learn to walk *quickly*—eventually running as quickly and smoothly as possible. In the *Setup* assignment, it is evaluated if you are able to train an agent at all (i.e. provide correct log files etc. as described below), while in the *Final Submission*, it is evaluated how well the agent has actually learned to walk and how quickly it has learned.

You are to write a short scientific report for the method, experimental results, and limitations in a provided LaTeX template that closely follows parts of the ICLR conference style guidelines. For the *Setup* assignment, you can simply submit the report template unchanged. You are to also provide a video of the agent and a log file as later explained. These files must all be zipped together, replacing the dummy username (`xxxx00`) with your CIS username (this is the same for both *Setup* and *Final Submission*):

```
xxxx00.zip
   xxxx00-agent-paper.pdf
   xxxx00-agent-code.ipynb (or .py)
   xxxx00-agent-code-hardcore.ipynb (or .py)
   xxxx00-agent-log.txt
   xxxx00-agent-log-hardcore.txt
   xxxx00-agent-video,episode=210,score=85.mp4
   xxxx00-agent-video-hardcore,episode=350,score=-92.mp4
```

You may not submit any additional files. To assist in this, the following template reports and starter code are provided to build on:

- [Reinforcement Learning Paper Template]
- [Coursework Template Starter Code]

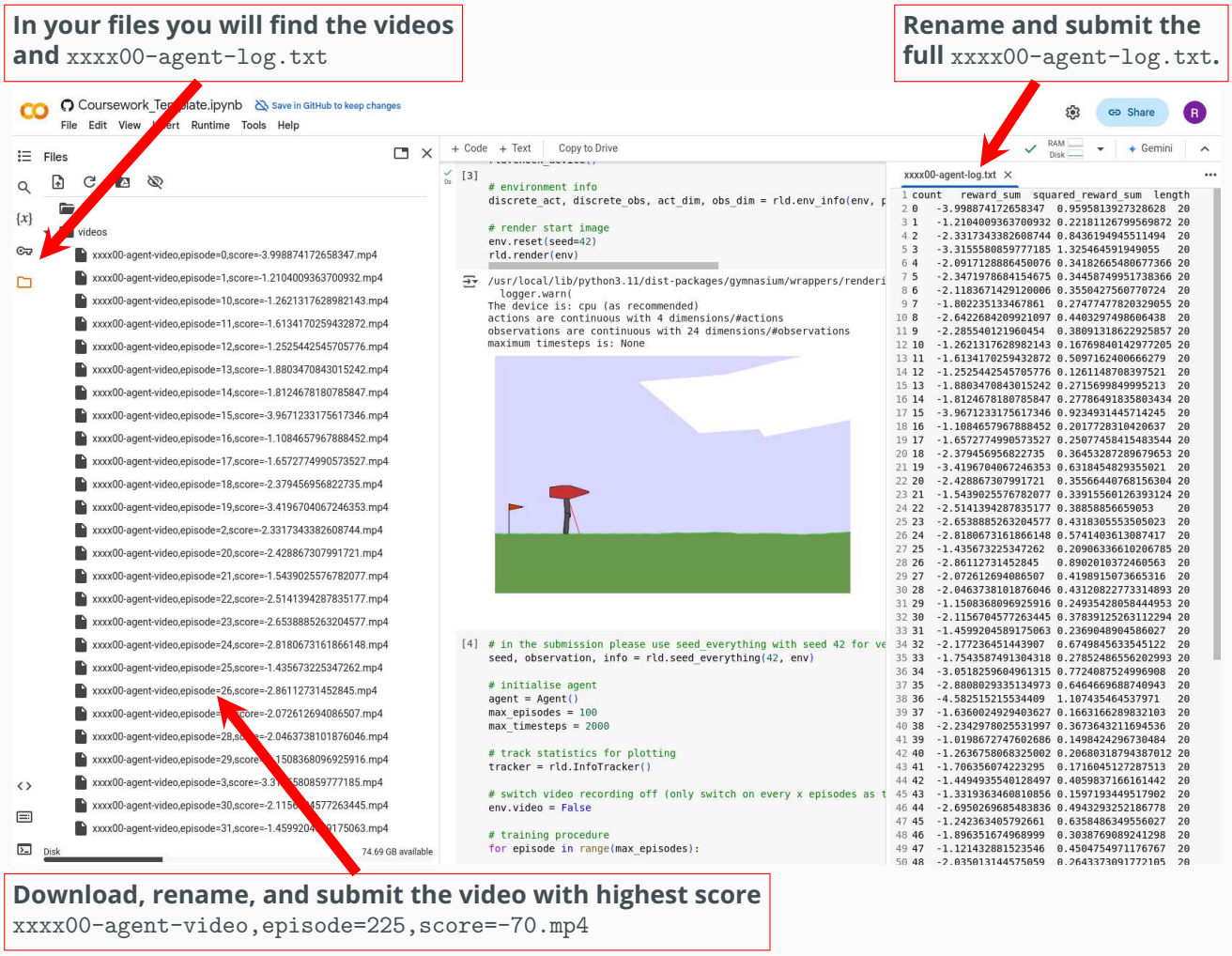
Note: The only difference between *Setup* and *Final Submission* is how these files are evaluated. For *Setup* you only need to have code for a basic training loop set up and running and provide correctly formatted log files and videos; your report can just be the template. For the *Final Submission*, we will evaluate the quality of your final report and how good your agent performs in the environments (see below).

Learning to walk efficiently

The task is to use deep reinforcement learning to train a small bipedal robot to learn to walk efficiently. This means you will be marked on how quickly you can train the agent to learn to walk by taking the smallest amount of environment steps possible. You will first develop and train your agent in an easy version of the `'rldurham/Walker'` environment (`hardcore=False`) and, in a second step, run it in a much more challenging version (`hardcore=True`). Both environments are based on the BipedalWaker environments provided by Gymnasium, which uses Box2D for its physics simulation. Use the previously linked code to get started. After your agent consistently gets scores over 300 and you are happy with how quickly it converges when trained from scratch in the easy version, you should transition to the more challenging one.

Submission

You must submit the full `xxxx00-agent-log.txt` and you must not change how its contents are formatted. The submitted video should show the highest score that you were able to successfully record within the **first 1000 training episodes (2000 for hardcore)**. Repeat this also for any experiments you have conducted in the hardcore environment. Recording a video is quite slow, so you may wish to do it every 25 or more episodes. Increasing this interval may slightly improve training performance, but you may miss recording a good episode. The following image explains how and what to submit:



You can submit an .ipynb or .py file. The code should be clear and well-structured. You do not need to strictly follow PEP-8 or any other guidelines for code quality with this assignment, but for each part of

the code it should be clear (from variable/function names, comments, or docstrings) what it does. If we find obscure code, for which it is not clear what it does and/or where it comes from, this may result in a penalty.

Restrictions

The log files must be collected in exactly the same way as the template code as we will run an automated script on them for assessing the algorithm convergence and score. You must submit **at least 1000 episodes (2000 for hardcore)** of log data. Please keep the max environment step count as 2000. The paper.pdf is restricted to a maximum of 4 pages (excluding references). Every page over this will incur a -10 mark penalty.

You can implement any RL algorithm that you like in your final solution as long as it is a *deep* reinforcement learning agent (it needs to have multiple layers).

Hardcore mode

After your agent consistently scores over 300 in the `Walker(hardcore=False)` environment and you're happy with its learning efficiency, train the agent on the `Walker(hardcore=True)` environment and submit a separate hardcore `video.mp4` and `log.txt` showcasing how well your agent performs in this much more difficult setting. We will examine how well the agent navigates the terrain in this setting. This is a tough environment, so don't be disheartened if the agent is unable to perform well in it. You can write about such experiments in your report, but make sure to also present the convergence results of the basic environment.

Using external code and generative AI

You are free to use external code and/or generative AI to solve the assignment. If you use any external code, you must explain in your report how you've used it (e.g. as starting point for your implementation or as a copy-paste drop-in for a specific part of your code etc.) and you must cite it (either the corresponding paper as literature reference, if one exists, or the GitHub repository or URL where you found the code as footnote).

If you use generative AI (such as ChatGPT) **in any way**, you must describe this in your report. This is not generally considered negative, on the contrary, it can be a positive aspect if used well. State what model(s) you have used (e.g. ChatGPT 5.1), what you have used it for (e.g. improving grammar in the report or getting suggestions for hyper-parameter tuning), and what your prompting strategy was. Be aware of the known shortcomings of generative AI (e.g. it tends to confirm rather than critique you; it tends to produce polished, good-looking output that may be void of meaningful content or simply be hallucinated; it tends to be good at standard tasks that appeared frequently in the training data, while failing in trivial edge cases).

Reinforcement learning marking scheme

The paper, code, videos, and log submissions will be marked as follows:

Setup (10%)

- **[1 mark]** Report
  - Was a report submitted (content may be unchanged from template)?
- **[1 mark]** Code
  - Was the code submitted?
- **[2 marks]** Log files
  - Are the log files correctly formatted?
  - Do they contain 1000 episodes for the easy environment and at least 100 for hardcore?
- **[2 marks]** Videos

  - Do the videos show the correct environments (easy and hardcore)?
  - Do the scores match those from the log files?
- **[4 marks]** Convergence
  - Does the agent improve in the easy environment?

Final Submission (90%)

- **[50 marks] Log files** (convergence and score)
  - Does the agent **explore efficiently** (get some good runs early on even if still unstable)?
  - Does the agent **converge robustly** (consistently get high scores towards the end with few bad runs)?
  - Does the agent achieve **high scores** (how good is the best run)?
  - Does the agent overall **learn efficiently** (the area under it's median curve is high)?
- **[25 marks] Report and implementation** (sophistication and mastery of the domain)

  Generally, we are looking for clear, accurate, and concise language, and a structured and rigorous approach in the experimentation.
  - What quality is the presentation of the underpinning theory?
  - Do the experimental results demonstrate scientific rigour?
  - How hackish is your implementation, or is it robust and well-designed?
  - Have you just cited and pasted code, or is there evidence of comprehension with further study and novel design extending beyond the lecture materials?
- **[15 marks] Videos** (intuition of final policy)
  - Is the agent walking fluently or with undesirable behaviour?
  - How fast is the agent running?
  - How effectively is it able to navigate the terrain?

Closing Comment

I hope that you enjoy this coursework and find watching your robot learn to walk rewarding! If you are struggling, please ask questions where we can discuss any issues, such as with programming or relevant theory.

Guidance

Before starting, here is some guidance for this environment:

1. I encourage starting with a modern method like TD3 (covered in lecture 8). The environment has a continuous action space (not suitable for DQN) and most of the classic continuous action space algorithms that work out-of-the-box on Pendulum-v0 (like SAC, PPO and DDPG) will need additional tricks and a lot of patience/tuning before showing any signs of convergence in the `Walker` environment.
2. Even a good agent for this environment can take hours to converge to a good policy. So plan for some long breaks while it's training. Note that the iterative process and long training times imply that you will benefit from starting early with this coursework to be able to complete it with good results.
3. It's common to implement a non-robust algorithm that sometimes works and other times doesn't due to poor initialisation and exploration (where the agent can get stuck in a minima, such as the robot always doing the splits and being unable to recover). If you've been training for several hours and your last few videos all have the same undesirable agent behaviour, this is an indicator that more exploration is needed. Try resetting and retraining the same agent and compare the logs/videos and see if it gets stuck doing the same undesirable behaviour again.
4. Unless you are prepared to spend time fiddling with vectorised environments, only train on the CPU (in Colab, go Runtime $>$ Change Runtime Type $>$ None)—unlike the DL model where you will want to use a GPU. This will give you more Colab time and you will likely be I/O bound by the `Walker` environment simulation where further transfer to GPU is often slower. In marking, we strongly favour environment sample efficiency (few steps) above parallelism and GPU occupancy.
5. I would not suggest trying to design and build a new agent for this completely from scratch (without reference code) unless you are reimplementing a paper that provides extensive technical details. You are allowed to re-implement and cite existing implementations that you have found on GitHub, and even re-use existing hyperparameters that people have found for this specific environment. But, by doing so, you must cite all author code you use and detail your own individual experiments in attempting to further improve their work in your report. Also, as stated in the marking scheme above, we do grade comprehension and novelty of the design, so implementations that mainly reproduce existing work will be held to higher standards for demonstrating understanding.

Frequently Asked Questions

How to install `rldurham`? Have a look at the instructions here: `https://github.com/robert-lieck/rldurham`. Make sure you follow those, as from experience, 9 out of 10 problems are either general Python problems or can be resolved by following these instructions.

Is reward scaling allowed? The short answer is: reward ***scaling*** **is allowed**, reward ***hacking*** **is not allowed**. In particular, a linear transform of the reward (i.e. adding a constant and/or multiplying by a constant) is OK and clipping to a fixed interval is also OK. A clear case of reward hacking would be to define helper rewards based on expert trajectories (e.g. training a model for a long time, then using this to bootstrap a new model "from scratch" by defining helper rewards or pre-training the policy to imitate expert behaviour), which is **not allowed**. In general, anything that incorporates explicit knowledge about final solution is **not** allowed. Anything that is not simple reward scaling/clipping is shaky ground and you probably should not do it.

What about intrinsic rewards? You **are allowed** to use intrinsic rewards, for instance, to encourage more structured exploration than simple random sampling. You are **not allowed** to make any use of knowledge about the final solution, which would be reward hacking as defined above.

What part of the template code can I modify? Generally, you should not modify how the environment is set up (mainly to avoid messing up logging), but you can change the sampling and learning part of the code to your needs.

Can I really just submit the template report PDF in the setup submission? Yes, in the setup submission, it is perfectly fine to submit the unmodified report template. Its content will not be evaluated. Submitting this file is only to ensure the setup and final submission consist of the exact same number and type of files to avoid confusion.

How to cite code I am using? In the final submission, you have to cite any code that you use in any way (except for standar Python packages, such as `torch` etc.). If there is a paper associated to the code, cite the paper and provide a link to the code itself e.g. the GitHub repo (you should use BibTeX for references to papers and can use footnotes for links). If there is only a GitHub repo, just reference that by providing the link.

Can I make modifications to my agent for the hardcore environment? Yes, you can – and probably should – have slightly different settings/agents for the easy and the hardcore environment, that is, you do not have to use the very same agent on both environments. However, you have to train the agent from scratch in the hardcore environment. That being said, it is a good idea to first tune hyperparameters on the easy environment before moving over to the hardcore environment (if the agent cannot even solve the easy one, it will be extremely hard to get it to work in hardcore).

I have found a perfect solution online, what should I do? How do you know it is perfect? Prove it by running thorough evaluations and showing that performance cannot be further increased, even when applying sophisticated modifications. There is almost certainly room for improvement!

Can I do transfer learning by pre-training on the easy environment before training on the hardcore one? You can tune hyperparameters etc. on the easy environment before attempting hardcore, but you have to train the agent from scratch in the hardcore environment to ensure comparability.

How should I trade off exploration and learning speed against robust convergence? As a rule of thumb, you should try to get some good runs as early as possible (typically requiring high exploration, which necessarily also results in some noise and bad outliers), while you should aim to minimise the number of bad runs towards the end (when approaching episode 1000/2000 for normal/hardcore, typically requiring exploration to approach zero to maximise robustness).

Can I include an appendix in the report? Yes, you can, but we may ignore it during marking. So anything important should go in the main text. You should only put supplementary information – such as additional figures or tables with hyperparameter settings etc. – in the appendix.