Reinforcement Learning Robustness and Adaptation in Continuous Control

CCE5502
University of Malta
Dr. Leander Grech

November 2025

1 Introduction

1.1 Repository Structure

The repository is designed to separate configuration from code. You will primarily interact with the `configs/` folder and the root-level scripts.

```
CCE5502-RL-Assignment-2025
├── configs/
│   ├── agent/
│   │   └── sb3_ppo.yaml
│   ├── env/
│   │   ├── ant.yaml
│   │   ├── crippled_ant.yaml
│   │   └── ...
│   ├── train.yaml
│   └── ...
├── src/
│   ├── agent/
│   │   └── ...
│   ├── env/
│   │   └── ...
│   └── ...
├── post_training_analysis.py
├── train.py
├── requirements.txt
└── README.md
```

You can clone/download it from here:
`https://github.com/leander-grech/CCE5502-RL-Assignment-2025`

Please take a look at the README file, which goes into more detail about the environment and RL algorithm we will be using in this assignment.

1.2 Hardware Note

You do NOT need a GPU for this assignment. Deep Reinforcement Learning (unlike Computer Vision) often bottlenecks on the CPU becuase the simulation (the environment) also runs on the CPU. For the network sizes and batch sizes used in this assignment (small MLPs), training on a modern CPU is often faster than moving data back and forth to a GPU.

Recommendation: Run everything on your laptop CPU. If you force usage of GPU, you might actually see slower training speeds.

1.3 Hydra

We use a tool called Hydra to manage our experiments. In RL, we have dozens on moving parts (environment settings, neural network architectures, learning rates, etc.). Hard-coding these in Python is both a nightmare, and not a good idea.

Hydra allows us to use YAML configuration files to swap parts in and out without changing the code. Here's how it works:

1. **Hierarchical Configs:** Look at `configs/train.yaml`. It has a `defaults` list.

   ```yaml
   defaults:
     - env: ant
     - agent: sb3_ppo
   ```

   This tells the script: "By default, load the standard Ant environment and the PPO agent."

2. **The Override Magic:** You don't need to edit files to change settings. You can change anything from the command line.

   - Want to change the learning rate?
     `python train.py agent.learning_rate=0.001`
   - Want to switch to the *Crippled Ant* environment?
     `python train.py env=crippled_ant`

1.4 Scripts & Arguments

You will use two main scripts. While Hydra handles the configuration, knowing the command-line arguments is essential.

1. `train.py`: This script initializes the environment and the PPO agent, then starts the training loop.

   - Usage: `python train.py [overrides]`
   - Key Overrides:
     - `task_name="my_run_name"`: Sets the folder name in `logs/runs`. **Always set this to keep your experiments organized**

  - `env=env_name`: Switches the environment (e.g. `env=ant` or `env=crippled_ant`).
  - `agent.learning_rate=float`: Overrides learning rate.
  - `agent.batch_size=int`: Overrides batch size.
- Example: `python train.py task_name="test_run" env=crippled_ant agent.learning_rate=3e-4`

2. `post_training_analysis.py`: This script loads the network weights of a saved RL agent and runs it for evaluation, plotting, or video rendering.
   - Usage: `python post_training_analysis.py --run [PATH_TO_RUN] [flags]`
   - Arguments:
     - `--run [PATH]`: **(Required)** Path to the specific run folder (e.g. `logs/runs/train_ant/2025-11-26_14-00-00`).
     - `--render`: If set, saves an MP4 video of the agent to the `videos/` folder.
     - `--plot`: If set, displays/saves the training reward curve.
     - `--cripple`: Forces the evaluation environment to be the `CrippledAnt`, even if the agent was trained on a *healthy* `Ant`.
   - Example: `python post_training_analysis.py --run logs/runs/my_run --render --cripple`

1.5 Installation

Before starting, create a new Python environment and install all dependencies, e.g. using conda: `conda create -n cce5502-rl-assignment python=3.9; pip install -r requirements.txt`

1.6 How to train your Agent

The main entry point in `train.py`

1. **The Default Run:** To train the standard Ant agent using the default PPO hyperparameters:

   `python train.py`

   This will create a folder in `logs/runs/train_Ant_default/...` containing your model checkpoints and Tensorboard logs.

2. **Naming Your Runs:** Always name your runs so you can find them later!

   `python train.py task_name="my_first_ant"`

3. **Changing Hyperparameters:** To train with a smaller batch size and higher learning rate:

   `python train.py agent.batch_size=64 agent.learning_rate=1e-3`

1.7 Evaluating Your Agent

Training produces numbers, but we need to see the agent to understand it.

1. **Generate Videos & Metrics:** Use the analysis script. You must point it to the specific run folder (the one containing `.hydra` and `.checkpoints`).

   ```
   python post_training_analysis.py --run
   logs/runs/train_Ant_default/2025-09-25_10-00-00 --render --plot
   ```

   - `--render`: Generates an MP4 video of your agent in the `videos/` folder.
   - `--plot`: Shows the training curves (rewards vs. timesteps).

2. **Cross-Evaluation:** You can take an agent trained in one environment and force it to be evaluated in another. Example: *Test your healthy Ant agent on the Crippled Ant environment:*

   ```
   python post_training_analysis.py --run logs/runs/train_Ant_default/...
   --cripple --render
   ```

2 Assignment

Submit a single ZIP file named `FirstName_LastName_CCE5502.zip` containing your PDF report and the `videos/` folder with agent recordings which solve the following tasks (Use the question number in the file name to indicate what it is referring to, e.g. **Q1_baseline_agent_Ant.mp4**). Do not include the `logs/` folder. To display training metrics, you can either download the CSVs and plot them, or take a screenshot from Tensorboard.

Q1: Establish a Training Baseline

Train a PPO agent on the standard `Ant-v5` environment using the default configuration provided in `configs/agent/sb3_ppo.yaml`.

- Run the training for 1,000,000 timesteps.
- **Deliverable:** A screenshot of the "Episodic Reward" curve from Tensorboard (or the plot generated by the analysis script). What is the maximum reward the agent achieves? Does it converge (flatten out)?
- **Deliverable:** A rendered video of your trained agent running. Describe the learned gait (e.g. dragging a leg, hopping, trotting). Are there any notable behaviours from the rendered video?

Q2: The "Reality Gap" (Zero-Shot Transfer)

Robots often break. We need to know what happens when our "optimal" policy meets an imperfect robot.

- Take the agent you trained in Q1. Using the `post_training_analysis.py` script with the `--cripple` flag, evaluate this healthy agent on the `CrippledAnt` environment (where one leg is disabled).
- **Deliverable:** The mean reward obtained in the crippled environment. Calculate the performance drop percentage compared to Q1.
- **Analysis:** Watch the video of the transfer and explain the new behaviour. Why does it fail if there are still 3 working legs?

Q3: Adaptation by Specialist Training

If zero-shot transfer fails, we must retrain. Train a **new** agent from scratch on the `CrippledAnt` environment.

- *Hint:* Use the command line override `env=crippled_ant`
- Train for the same number of timesteps (1M).

- **Analysis:** Compare the learning curve of this specialist agent to the baseline agent from Q1. Is it harder to learn to walk with a broken leg? Comment on the final reward, learning speed, and quality of rendered episodes from the specialist agent(s) you have trained.
- **Deliverable:** A video of the specialist agent. How does its gait differ from the baseline agent?

Q4: The Hyperparameter Hunt

RL is sensitive to hyperparameters. The `CrippledAnt` task is harder, so the default settings might not be optimal.

- Perform a small "grid search" by training two additional agents on `CrippledAnt` with different settings:
  1. **High Learning Rate:** `agent.learning_rate=1e-3`
  2. **Small Batch Size:** `agent.batch_size=64`
  3. *Feel free to document any other set of parameters which performed better.*
- **Analysis:** Plot the training curves of these two new runs against your Q3 baseline. Which setting results in the most stable learning? Did any settings cause the agent to fail completely? If your new agent trained better than the baseline, try to comment on the reasons of this performance boost.

Q5: Generalization vs. Severity

The `CrippledAnt` environment has an `injury` parameter that controls how many legs are affected (options: `little`, `medium`, `hard`).

- Modify the configuration (or use CLI overrides) to train an agent on the "hard" injury setting.
- **Analysis:** Plot the training curves and compare the new agent to the Q1 baseline and the best Q4 agent obtained. Compare the gait of the new agent to that of agents trained on different injury levels. Has the agent discovered a fundamentally different mode of locomotion (e.g. crawling vs. hopping) to cope with severe damage?
- **Deliverable:** A video of the best agent trained on `CrippledAnt` with a *hard* injury setting.