# Assignment # 4 - Continuous Control through Deep Policy Gradient Methods

Author: Prabhat Nagarajan CMPUT 628, Deep RL: Winter 2025

## Instructions and General Notes

- **Due Date**: March 28, 2025.
- **Late Policy**: See syllabus.
- **Marking**: There will be 150 total marks.

### Collaboration and Cheating

Do not cheat. You are graduate students. You can discuss the assignment with other students, but do not share code with one another. Do not use language models to produce your solutions.

### General Advice

We highly recommend **starting the assignments early**. In machine learning and reinforcement learning, training agents or learning systems requires a different skillset from traditional programming and software engineering. Oftentimes the process of debugging is far more time-consuming, as code may be only "somewhat" wrong and so discovering why your agent is underperforming is not a straightforward task. Moreover, debugging often involves having agents run for periods of time, and training can be both expensive in terms of both time and compute. Building agents is also not merely validating program correctness, but involves a lot of trial and error on the programmer's part (not just the agents).

### Software and Dependencies

In this assignment, you have more freedom.

This assignment will use:

- Python 3.11
- `numpy`
- `matplotlib`
- `gymnasium[mujoco-py]`
- One amongst the following:
  - `jax`, `flax`, `optax`

- `torch`,
- `tensorflow`,

### Submission

Please submit a zipped folder titled `a4_<ccid>.zip`, where you replace <ccid> with (can you guess?) your ccid. E.g., a4_machado.zip. The unzipped folder should be `a4_<ccid>`. **Failure to do so will cost you 5 marks.** This zip file should contain:

- pdf titled a4_<ccid>.pdf
- All the python files from a4.zip containing your solutions in `a4_<ccid>.zip`

## Assignment

The previous assignments have been focused on value-based reinforcement learning approaches. In this assignment, you will implement a policy gradient approach. Specifically, you will implement one of five algorithms:

- Deep Deterministic Policy Gradient (DDPG)
- Twin-delayed Double DDPG (TD3)
- Soft-Actor Critic (SAC)
- Trust Region Policy Optimization (TRPO)
- Proximal Policy Optimization (PPO)

You will select *one* of these five to implement. In particular you will apply your algorithm to two continuous control tasks from Gymnasium:

- Ant-v4
- Walker2d-v4

### More details, Performance Requirements, and Report Requirements [80 marks]

In this part of the assignment, you must select one of the aforementioned five algorithms, and implement it from scratch. You may not build-off of open-source libraries or implementations. That said, while you must build your implementation (including training loops, etc.) you may consult or reference implementations online. When we say implement "from scratch", you are permitted to reuse code from previous assignments, if you should choose, though it is not a requirement.

You do not necessarily have to reproduce these algorithms 100% faithfully. Your implementation should strongly resemble your algorithm of choice, but you may make minor modifications. Not everything has to be by the book (or by the paper?). If you do so, they should be described

in your report. For example, maybe TD3 works better in your environment without clipped Double Q-learning. It would be fine to implement it this way even though that makes the algorithm technically not “TD3”.

**Marking & Requirements**

- Describe your full solution to each task in terms of algorithm, exploration mechanisms, hyperparameter choices, etc. If any atypical changes are made to the base algorithm, describe it. No more than 3/4 of a page. **[30 marks]**
- For Ant-v4, marking is as follows, depending on the algorithm you choose. **[25 marks]**
  - **DDPG**: Score of 700 earns 10/25 marks. Anything less than that earns you no marks. Score of 900 consistently earns full marks. All runs must be less than or equal to 1M timesteps.
    - **Warning: DDPG is known to be unstable on Ant-v4. Finding the correct hyperparameters or integrating components of TD3 may boost performance.**
  - **TD3/SAC**: Score of 2000 earns 10/25 marks. Anything less than that earns you no marks. Score of 3500 consistently earns full marks. All runs must be less than or equal to 1M timesteps.
  - **TRPO** Score of 600 earns 10/25 marks. Anything less earns you no marks. Score of 1000 consistently earns full marks. All runs must be less than or equal to 2M timesteps.
    - **Warning: We have found that TRPO performs quite poorly at this task within 2M timesteps. We can indeed reach this required performance, but we were not able to do much better.**
  - **PPO** Score 1500 earns 10/25 marks. Anything less earns you no marks. Score of 2250 consistently earns full marks. All runs must be less than or equal to 2M timesteps.
- For Walker2d-v4, marking is as follows, depending on the algorithm you choose. **[25 marks]**
  - **DDPG**: Score of 1250 earns 10/25 marks. Anything less than that earns you no marks. Score of 1750 consistently earns full marks. All runs must be less than or equal to 1M timesteps.
  - **TD3/SAC**: Score of 2000 earns 10/25 marks. Anything less than that earns you no marks. Score of 3500 consistently earns full marks. All runs must be less than or equal to 1M timesteps.
  - **TRPO/PPO**: Score 1500 earns 10/25 marks. Anything less earns you no marks. Score of 2500 consistently earns full marks. All runs must be less than or equal to 2M timesteps.

- **For each environment, provide a plot with the x-axis showing timesteps and the y-axis showing the undiscounted return. Describe in your report how you produce this curve and what a point on this curves means. Show the mean performance across at least three seeds, and in a lighter shade, include the learning curves of the individual runs.**
- **Your report must include the commands needed to reproduce your figures for each task.**
- *****Note on "consistently": Above, when we say your algorithm must score a certain amount "consistently", we mean that the mean curve must cross this threshold at some point within the last 20% of training.**

This assignment is an opportunity for you to play around with the algorithms you might want to use for your own research, or develop a good understanding of for future research.

**Suggestions**

- Look at open source implementations and the research papers for good architectures and hyperparameters.
- Look at the appendices of papers to find key implementation details.
- Try getting your agent working on a simple task like MountainCar, Reacher, or Inverted-Pendulum
- Figure out how to get a pulse on whether your training run is working effectively. Most of these algorithms have some "typical" learning curves in these environments. You expect the learning curve to exhibit some behavior (e.g., a certain minimum score after a certain number of training steps). This can be used to fail fast and identify issues in a timely manner.

## Science [70 marks]

In the second part of this assignment, you will perform an experiment on the algorithm you implemented to try and understand it better. Insightful results may even lead to a workshop or conference paper. In particular, you will design an experiment to study an aspect of your algorithm that you choose on at least two of the environments above.

There are many ways to study algorithms, but we will restrict ourselves to four broad types of experiments (and we take a generous interpretation of these 4 types of experiments):

- **Measure an effect.** In this type of experiment, you might measure the effect or impact of some aspect of your algorithm by collecting metrics that help you understand something. For example, maybe you want to measure how fast the selected actions change. Perhaps you want to measure the entropy of a policy over time.

- **Compare or ablate a component of an algorithm.** If some algorithm or paper makes an implementation decision, design choice, or introduces algorithmic subcomponent, we can gain a better understanding of its importance through an ablation. For example, if studying DQN, one can ablate target networks by running the algorithm with and without target networks. Another generic example is normalization schemes. There are different normalization schemes, and we can ablate algorithms by observing them with and without normalization, or by comparing normalization schemes.
- **Consider alternatives**. Sometimes components of algorithms cannot be ablated. E.g., there are interdependencies between some feature and other parts of the algorithm. E.g., you cannot have an actor-critic algorithm without a critic. But in many cases you can try an alternative approach to one from the algorithm to understand its importance.
- **Try to augment or improve an algorithm.** Perhaps you are ambitious, and are not simply trying to understand components of existing algorithms (though a noble pursuit), and believe you understand an algorithm's limitations or mechanisms well enough to propose an improvement. Feel free to try this. We will restrict you to augmenting algorithms, though, rather than defining entirely new ones, as defining new algorithms has a much broader scope.

**Marking** For this experiment you must:

- **Motivate the experiment**: This is not necessarily for a research paper, you do not need any justification to investigate something beyond being curious. However, try to elaborate, why are you curious? For example, is there not much literature on alternative approaches so you want to try one that seems sensible? Why might that alternative be sensible? Did you think some hyperparameter or aspect of the algorithm may be unimportant so you want to find out whether or not it is important? Are you curious why they chose that? Did you read another paper that suggests something contrary to what you see here? Your response can be anywhere from a few sentences to no more than half a page. **[5 marks]**
- **Describe the object (phenomena, algorithmic detail, mathematical detail, metrics such as loss/entropy/overestimation) that you are trying to study**. For example, explain the technical details of how this manifests in the algorithm. If you are studying soft target updates, write a short explanation of what soft target updates are and how they are done in the algorithm that you are working with. Provide context so a reader with basic knowledge of RL can understand. Describe very clearly the algorithmic or implementation detail that you are testing. Your response may be a few sentences to 3/4 of a page. Your response will be marked on quality of exposition. **[15 marks]**
- **Describe the experiment**, and what you are trying to study. Describe how you will draw conclusions based on your experimental design. Tell us your hypothesis. It is fine to run follow-up experiments based on what you see. *Marks will be given not on the quality of*

*experimental design, but rather the clarity in describing the experiment. Are you clearly describing what you are measuring, how your experiment is conducted, etc.?* **[15 marks]**

- **Share your results**. Describe your plots and results clearly, so an independent reader can draw their own conclusions. *You are marked on how clear you are in describing your results. Your reader should not have to mind-read what your plot is trying to show when you can provide a description of the plot. Also, describe literally what you see in your plots/tables etc.* **[15 marks]**
- **Share your analysis.** Then provide your own analysis and interpretation of the results, including any insights, if any. Note that this is not peer-review, which means your results do not have to be "interesting" or conclusive. Depending on your results, we acknowledge that you may have more or less to say. *Your marks here are based on what you are able to glean conditioned on your experimental results*. **[10 marks]**
- **Describe the limitations** of your study and what you would do to improve it. Are there better ways to study the object/phenomena you are trying to study than the experiment you chose? Hypothetically, what could you do to be more sure of your results? *Here is where you will be marked on the quality of your experimental design. You were marked on the experimental design for clarity, here you are marked on your ability to find issues with your own experiments.* **[10 marks]**
- **Optional: Appendix** if you want to go above and beyond in your report (without it being marked), you can add an Appendix that may or may not be read and may or may not influence your marking. It is up to the professor's discretion.

You may or may not learn a lot from the experiment, which is why you are doing it in the first place. The recommendation is to try and go granular with your experiments. For example, rather than simply measure the impact of different replay buffer sizes on performance, can you measure some quantity that you think will correlate with replay buffer size?