Deep generative models and learning to walk

Deep learning and reinforcement learning summative assignment

Michaelmas term, 2021

Introduction

The assignment has two parts: (1) Deep Learning (50 marks) and (2) Reinforcement Learning (50 marks) accordingly, for a combined total of 100 marks available.

In short, the deep learning part of the assignment is to build a generative model that creates unique and diverse images subject to some criteria. The second part of the assignment is to train a bipedal robot to learn to walk in a 2D physics simulation.

For each part, 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 reinforcement learning part, you are to also provide a video of the agent and a log file as later explained. These files must all be zipped together like this. You may not submit additional source code files:

```
submission.zip
   part1-model-paper.pdf
   part1-model-code.ipynb (or .py)
   part2-agent-paper.pdf
   part2-agent-code.ipynb (or .py)
   part2-agent-video,episode=210,score=-85.mp4
   part2-agent-log.txt
```

To assist in this, the following template reports and starter code are provided to build on:

- [Deep Learning Paper Template] - *login with durham email, on 'overleaf.com' click 'make a copy' to edit*
- [Reinforcement Learning Paper Template]
- [Part 1: Google Colab Generative Model Starter Code]
- [Part 2: Google Colab Bipedal Walker Starter Code]

Part 1: Deep generative model

Using the CIFAR-10 dataset and/or a high-resolution dataset such as STL-10, individual classes of LSUN or the FFHQ dataset, train a deep generative model to synthesise unique images that will be judged on their realism, diversity and uniqueness from the original training data. Then write a short paper up to a maximum of 4 pages using the provided LaTeX template, writing up the methodology, results, and limitations of your approach alongside a short abstract.

The methodology should explain the underpinning theory of your model formally, and the results section should show: (1) a unique batch of 64 non cherry-picked samples (2) interpolations between 8 pairs of your samples, and (3) some cherry-picked examples (a selection of the best images you have seen your model generate).

The report should be written like an academic paper, with formal mathematical notation that should try to follow the ICLR guidelines (see the template for more information). Therefore your discussions should be short, clear, and concise —*less is more*. Where appropriate, it is recommended to include a high-level architectural diagram in the paper to help explain your approach.

You can use any generative model architecture that you like, and you can use any sampling strategy. However, there are penalties and bonuses that will influence your design, summarised in this table:

Use any adversarial training (e.g. GAN) method	-4 marks
Train only on CIFAR-10	-2 marks
Train with STL-10, LSUN or FFHQ resized to 48x48 pixels	$+1$ mark
Train with STL-10, LSUN or FFHQ resized to 64x64 pixels	$+1$ mark
Train with STL-10, LSUN or FFHQ at 96x96 pixels or more	$+2$ marks
Manually edit (paint) any images or outputs	-50 marks
Use or modify someones code without referencing it	-50 marks
Use pre-trained weights from another model	-50 marks
Every page over 4 pages in the paper (excluding references)	-5 marks

Table 1: Penalties and bonuses stack, and are added onto the final mark.

Please state at the end of the paper the total bonus or penalty you are expecting to receive according to this table. For example, if you successfully train a GAN using data both from CIFAR-10 and from STL-10 resized at 64x64, you can expect to receive an overall -2 mark penalty because: -4 marks (as its a GAN), then 1+1 = +2 marks for STL-10 at 64x64 resolution. If you submit a paper that is 7 pages long, you will receive an additional -15 mark penalty.

Generative model marking scheme

The paper and submitted code will be marked as follows:

- **[20 marks]** Scientific quality and mathematical rigor of the paper and solution
  - Strategy and presentation of the underpinning mathematical theory
  - Architectural design, sophistication, appropriateness and novelty
  - Clarity, simplicity and frugality of both the scientific writing and the implementation
- **[15 marks]** Realism of the samples
  - Is the sampled batch of images blurry?
  - Do the image objects have realistic shapes and textures? Do they look real?
  - Do the interpolations look like linear alpha blendings or are all midpoints realistic?
- **[15 marks]** Diversity and uniqueness of the sampled batch of model outputs
  - How different are the images from their nearest neighbours in the dataset?
  - How diverse are the samples within the batch of 64 provided?
  - Do all the images look similar? Is there any mode collapse?

PyTorch Training

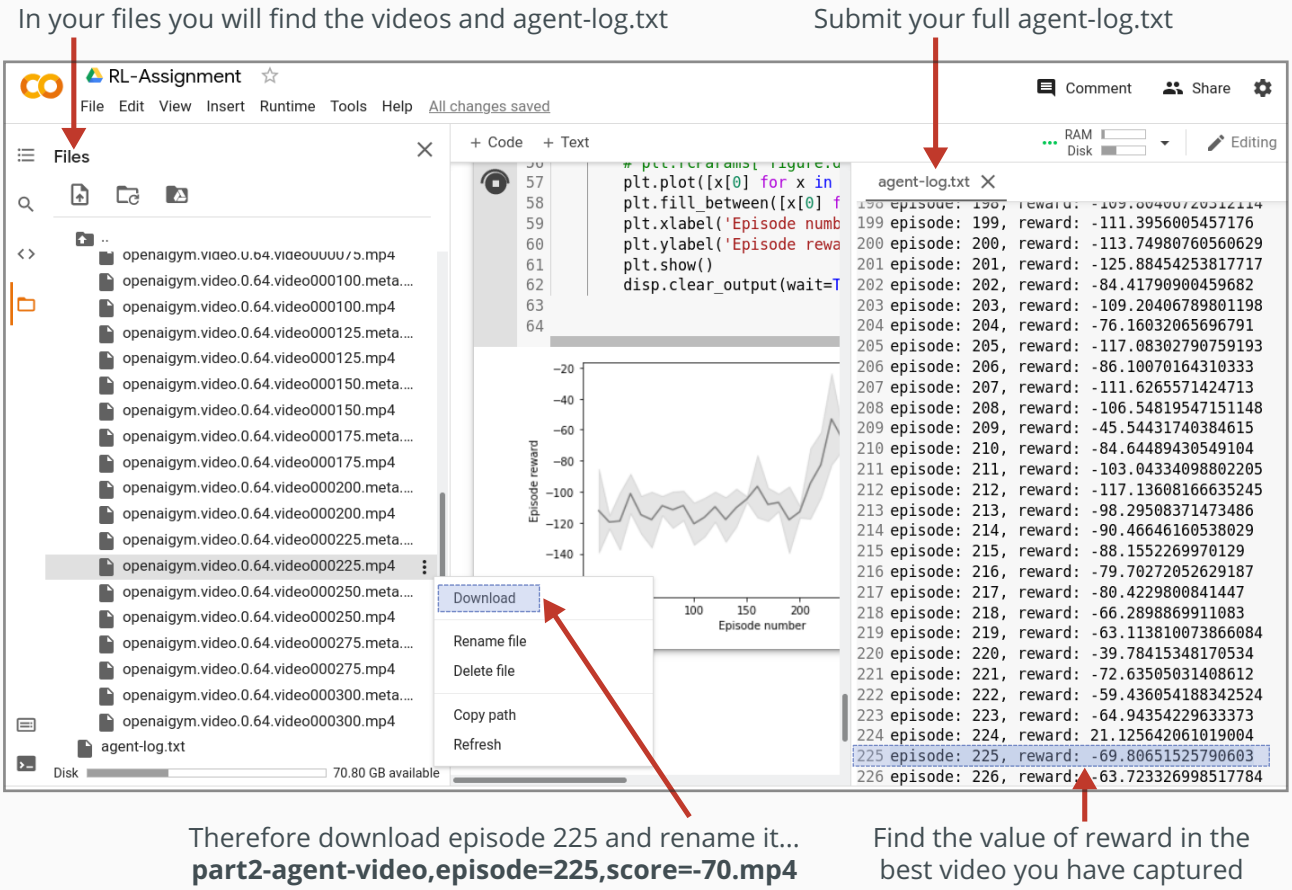
This assignment can be completed entirely using Google Colab, or you may wish to register for an account and train on NCC: `http://ncc.clients.dur.ac.uk/` (only available on the internal university network). Users without remote server experience or job queuing system experience (such as SLURM) are recommended to continue to use Google Colab, which is just as fast for PyTorch training. If using NCC, please carefully read the documentation and respect other users on the job queuing system.

Part 2: 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. The BipedalWalker-v3 environment for this is provided by OpenAI Gym, which uses Box2D for its physics simulation. Use the previously linked code to get started.

Submission

You must submit the full `agent-log.txt` and you must not change how its contents are formatted. The submitted video should show the highest score that you are able to successfully record. 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:



The code submission must be written with clarity and minimalism. You can submit an .ipynb or .py file. You do not need to follow PEP-8 or any departmental guidelines for code quality with this assignment. Try to keep comments to a minimum, as good code should speak for itself.

Restrictions

The log file must be collected in the same way as the template code as I will run an automated marking script on it for assessing the algorithm convergence and score. You must submit at least 500 episodes of log data (unless your agent consistently reaches scores around 300 in fewer than 500 episodes). Please keep the max environment step count as 2000. The reinforcement learning paper is restricted to a maximum of 4 pages (excluding references). Every page over this will incur a -5 mark penalty.

You can implement any RL algorithm that you like in your final solution. You do not necessarily have to use backpropagation, however you should use *deep* reinforcement learning (so the agent must be deep in the sense that it needs to have multiple layers).

Hardcore mode

If you feel you have aced the basic environment and need something more challenging to show the capabilities of your agent, you may also train in the BipedalWalkerHardcore-v3 environment and submit a second video e.g. `part2-agent-video-hardcore,episode=610,score=185.mp4` showcasing how well your agent can perform in this more difficult setting. You can write about such experiments in your report, but make sure you also present the convergence results of the basic environment.

Reinforcement learning marking scheme

The paper, code, video, and log submission will be marked as follows:

- **[25 marks]** Convergence and score
  - Does the agent consistently get high scores?
  - How many training episodes are needed to get high scores?
  - Is the algorithm stable? Will it always converge or does it sometimes get stuck?
- **[15 marks]** Sophistication and mastery of the domain
  - 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 their evidence of comprehension with further study and novel design extending beyond the lecture materials?
- **[10 marks]** Intuition of video
  - Is the agent walking or stuck with undesirable behaviour?
  - How fast is the agent running?
  - How effective is it able to navigate the terrain?

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 BipedalWalker-v3.
2. Even a good agent for this environment will take 1-2 hours training before it starts to show signs of learning anything sensible, and perhaps a few more hours before it starts to get non-negative scores and walk forwards. So leave Colab running and take a long break when its training.
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 Colab and retraining the same agent for another hour 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 the BipedalWalker-v3 environment simulation where further transfer to GPU is often slower. In marking, I will 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.

Closing Comment

I hope that you enjoy this coursework. If you are struggling, please ask questions where we can discuss any issues, such as with programming or relevant theory.