EEC4400 Assignment (Sem 1 AY25/26)

1 Exploring Reinforcement Learning: Q-Learning, Naïve DQN, DQN and DDQN for Cart-Pole

Reinforcement Learning (RL) is a fundamental branch of machine learning where agents learn to make sequential decisions by interacting with an environment. Unlike supervised learning which relies on labelled data, RL agents improve their performance through trial and error, receiving rewards based on their actions. One of the widely used environments for testing RL algorithms is **Cart-Pole**, a classic control problem that aims to balance a pole on a moving cart by applying left or right forces.

In deep reinforcement learning (DRL), neural networks (NNs) are used to approximate or store the key components such as the Q-values in Q-Learning. In this assignment, students will implement and explore four DRL approaches (i.e. referred to as the "DRL algorithms") to solve the Cart-Pole environment:

1. **Q-Learning with Neural Network (Q-Network)**
   A direct application of Q-learning, where a neural network approximates Q-values without a replay buffer or target network.

2. **Naïve DQN (Single Network)**
   A baseline approach using a single neural network for both selecting and evaluating actions.

3. **DQN (with Target Network)**
   This is the standard DQN [https://arxiv.org/abs/1312.5602] approach that features a periodically updated **target network** to stabilize training.

4. **Double DQN (DDQN)**
   An enhanced method that addresses overestimation bias by decoupling action selection from action evaluation [https://arxiv.org/abs/1509.06461].

Through this assignment, students will gain hands-on experience in implementing different DRL techniques, compare their strengths and limitations, and understand how different hyperparameters can impact the DRL agent's performance in sequential decision-making tasks.

2 Platform

To facilitate streamlining of implementation and ease of grading, you are expected to use a Jupyter notebook to implement the DRL algorithms.

A skeleton code notebook file is provided which you must use to complete the assignment by following the guidance in the file. Some useful aspects of the assignment are already coded, and their functionalities need to be built upon to complete the assignment.

3 Program Implementation

This guide outlines the structure of the skeleton notebook, highlighting the parts students need to implement.

Note: You are not allowed to use reinforcement learning libraries like stable baselines, RLlib etc. for this assignment.

3.1 Background – [Full Code Provided]

This section introduces the Cart-Pole environment and guides you through basic Gymnasium operations. Study the code to familiarize yourself with the environment before implementing the DRL algorithms.

3.2 Configuring TensorBoard – [Full Code Provided]

This section involves creating the log directory ("eec4400_logs") to store the training runs of each algorithm to visualize in TensorBoard.

3.3 Training and Evaluating DRL Agents

Each student is responsible for implementing and training one of the following DRL algorithms:

- **Q-Network** (Student 1) ^
- **Naïve DQN** (Student 2) ^
- **DQN** (Student 3) ^
- **DDQN** (Student 4) ^

The general workflow for training and evaluating each DRL algorithm follows these steps:

1. Initialize baseline hyperparameters (see section 3.5)
2. Construct the neural network + [write code]
3. Implement, train and evaluate (see section 3.4) the four DRL agents [write code]
4. Integrate TensorBoard for tracking training progress
5. Compute and plot the **moving average training reward** with *window_size = 20* [Student1: write code]
6. Compute and plot the **evaluation reward mean** and **variance** per episode. [Student1: write code]
7. Track and report the **average training time** per episode
8. Repeat steps 2, 3, 5, 6, 7 for alternative hyperparameters (see section 3.5).

^ All four students need to discuss and understand the similarities and differences between the four DRL algorithms in order to achieve a streamlined and consistent flow of program statements and usage of variables and data structures in the implementation of the four DRL algorithms.

3.4 Performance Evaluation of a DRL policy

At the end of each training episode, the DRL policy at that point is evaluated by using it to generate actions for a few episodes and observing mean and variance of the rewards obtained. The performance of a DRL agent can be seen from the **training reward**, **evaluation reward mean** and **evaluation reward variance** of each episode as training progresses. The average training time per episode is also calculated at the end of training.

Plot the moving average training reward, evaluation reward mean and evaluation reward variance vs episodes for the DRL policy being studied.

3.5 Hyperparameter Tuning

Hyperparameter tuning is essential for optimizing model and algorithm performance and stability.

In deep reinforcement learning (DRL), we can regard the hyperparameters as belonging to two groups: the first group controls the performance of the neural network model (let us refer to these as the "NN hyperparameters", which includes the network structure) and the second group controls the performance of the DRL agent, given a particular neural network model (let us refer to these as the "RL hyperparameters").

Identify key hyperparameters and determine which are NN hyperparameters and which are RL hyperparameters. Write these in your report (see section 4).

The whole group should discuss and agree on a set of NN hyperparameters with reasonable values (let us refer to this as NN-hp-set1) and a set of RL hyperparameters with reasonable values (let us refer to this as RL-hp-set1) as the starting point.

Each student runs the experiment with his/her designated DRL algorithm with hyperparameters NN-hp-set1 and RL-hp-set1 to obtain the **baseline policy** for that DRL algorithm.
Evaluate the performance of the **baseline policy** (see section 3.4).

After observing the DRL results and analysing the internal NN weights using TensorBoard (see section 3.6) for the **baseline policy**, the whole group collectively designs a new set of NN hyperparameters (let us refer to this as NN-hp-set2) and a new set of RL hyperparameters (let us refer to this as RL-hp-set2), with the aim of improving the performance obtained so far. Describe your considerations in your report (see section 4).

Note: In practice, hyperparameter tuning is done programmatically using procedures like random search, grid search or sequential model-based optimization (SMBO) etc. For simplicity here, we skip these procedures and do 1-step manual tuning.

Each student then runs one more experiment with his/her designated DRL algorithm with hyperparameters {NN-hp-set2, RL-hp-set2} to arrive at an **alternative policy**.
Evaluate the performance of the **alternative policy** (see section 3.4). Analyse the internal NN weights using TensorBoard (see section 3.6) for the **alternative policy**.

Note: Use the same NN and RL hyperparameter values and neural network architectures across all four DRL algorithms in order to get a fair comparison of the baseline and alternative policies produced by the different DRL algorithms.

3.6 Plot Reward Graphs across DRL algorithms

Plot the moving average training reward vs episodes of the better DRL policy (baseline or alternative policy) for all the four DRL algorithms Q-Network, Naïve DQN, DQN and DDQN on the same graph.

3.7 Visualization with TensorBoard – [Full Code Provided]

After the various DRL algorithms and models are trained and evaluated, the collected statistics and resulting neural networks can be inspected using TensorBoard. The callback function (refer to the skeleton code file for more information) used when training the agents stores information collected during the runs in the log directory. This information can be visualized using TensorBoard.
Note: There will be different results seen using TensorBoard for the different policies produced by the different DRL algorithms.

3.8 Extra Exploration

Each student may perform a little extra exploration or enhancement related to his/her part of the work.
Place this work in the individual section of the Jupyter notebook and describe it in the individual section of the report.

4 Results and Report Writing

The experiments above provide two versions for each DRL algorithm:

- **Baseline DRL policy with baseline NN**: trained with parameters NN-hp-set1 and RL-hp-set1.
- **Alternative DRL policy with alternative NN:** trained with parameters NN-hp-set2 and RL-hp-set2.

Collect all the required plots (including TensorBoard visualizations) and performance metrics.

4.1 Evaluation within a DRL algorithm

4.1.1 Different NN structure and hyperparameters

Tabulate the values of NN-hp-set1 and NN-hp-set2.

Explain the considerations behind the choice of neural network structure and each hyperparameter value for the baseline NN and alternative NN and comment whether the desired effect was achieved.

Examine the baseline NN and alternative NN.
Using the statistics and results captured by TensorBoard, comment on the loss or error vs iterations in each case.
Examine and comment on the magnitudes of the internal parameters or weights at different parts of the NN.

4.1.2 Different DRL hyperparameters

Tabulate the values of RL-hp-set1 and RL-hp-set2.

Explain the considerations behind the choice of each hyperparameter value for the baseline DRL policy and alternative DRL policy and comment whether the desired effect was achieved.

Using the statistics and results captured by TensorBoard, comment on the learnt policies stored in the NNs.
By manually presenting certain observations as the input of the appropriate NN, examine and comment on the output Q-values.

The performance of the two DRL policies can be seen from the moving-average training reward, evaluation reward mean and evaluation reward variance vs episodes, whose graphs were plotted in section 3.4. The average training time per episode for the two DRL policies were also calculated.
Include these graphs and values in the report.
Comment on these observations.

4.2 Evaluation across DRL algorithms

4.2.1 Performance of different DRL algorithms

In section 3.6, the moving average training rewards vs episodes of the better DRL policy for all the four DRL algorithms Q-Network, Naïve DQN, DQN and DDQN have been plotted on the same graph.
Include this graph in the report.

Compare the performance of the policies learnt through these algorithms in terms of the task effectiveness for Cart-Pole, speed of convergence and stability. Explain why the different algorithms can bring about these differences.

How does experience replay (Q-Network vs Naïve DQN), target network (Naïve DQB vs DQN), and decoupling action selection (DQN vs DDQN) affect model performance?

4.2.2 Average training time per episode of different DRL algorithms

Tabulate and compare the average training time per episode of the four selected policies in the previous section.

Is the average training time per episode a good performance metric of the DRL algorithms?
Explain why or why not.

Which algorithm is computationally more expensive, and is the performance gain worth the cost? Why?

4.3 Report Writing

Write a short report of no more than 10 pages highlighting your work in this assignment. The report should contain the following information to get full marks:

- The reasoning behind the design of the neural network architectures and DRL agents.
- Answers to the questions in sections 4.1 and 4.2 [#]
- Any extra efforts can be highlighted in the report, but make sure you adhere to the page limit.
- Include the Statement of Contributions at the end of the report.
- Cite your reference sources, if any. The reference list does not count towards the 10-page limit.

[#] Keep your answers short and to the point. Use the experimental results and plots to support your claims. Any reasoning provided without appropriate results to back it up will be considered invalid.

5 Division of Responsibilities

The tasks to be completed by each student or the entire group are summarized here:

	Task	Student			
		1	2	3	4
Jupyter	Construct Q-Network agent and evaluate performance	✓			
	Construct Naïve DQN agent and evaluate performance		✓		
	Construct DQN agent and evaluate performance			✓	
	Construct DDQN agent and evaluate performance				✓
	Design 2 sets of hyperparameters for the four agents[#]	all			
	Implement moving average and other plotting functions	✓			
	Plot reward vs episodes of the four DRL agents for comparison	all			
	Visualization with TensorBoard	✓	✓	✓	✓
Report	4.1.1[#], 4.1.2[#]	✓	✓	✓	✓
	4.2.1, 4.2.2	all			
	[#]Reasoning behind the design of the NN structures and DRL agents	all			

all: collaborated item. ✓: individual item.

Apart from the tasks stated in the table above, you are welcome to write extra code to collect additional experiment results to validate your analysis done in the report. If such code is included, please organize the experiment code by person, and indicate each student's work clearly.

6 Submission and Grading

6.1 Submission

Submit your completed Jupyter notebook with all code, output and results (which should be named as "EEC4400-Groupxx.ipynb", where xx is your group no.) and the report in PDF format with file name "EEC4400-Groupxx-Report.pdf" by zipping both the files as **EEC4400-Groupxx.zip** and uploading this file* by **11.59pm on Saturday 15 November 2025.** There should be only 1 submission from each group.
**Note*: Uploading instructions will be given nearer the deadline.

6.2 Grading

- DDQN has not been covered in class, hence, students are expected to do their own research on the topic.
- Students within the same group may obtain different scores.
- Individual tasks will be graded independently, i.e., the training and performance evaluation of the DRL agent constructed by student 1 will not affect the grading of students 2, 3 and 4, and vice versa.
- DRL agent construction is a complicated task, and there are many aspects to be considered for solving a specific real-life problem. You are expected to make use of the concepts learnt in class and some research (if necessary) to support your choices made.
- The performance of the DRL agents will not be the main factor considered in grading; however, at least some of the agents should perform reasonably well. Each student may perform some extra exploration or enhancement related to his/her part of the work.
- For the report, students should not spend too much effort summarizing or explaining the code. Focus on analysis of results and explanation of observations.