BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

WORK INTEGRATED LEARNING PROGRAMMES DIVISION

Deep Reinforcement Learning

Lab Assignment 2

Total Marks = 13 Marks (7M for the first part and 6M for the second part)

Intended Learning Outcome:

Students should be able to

- Understand the basic functionality of DQN, DDQN, and Actor Critic.
- Implement the concepts of DQN, DDQN, and Actor Critic.

Prerequisite:

(1) Students should go through the lectures CS9, CS10, CS11, CS12, CS13 ;

(2) Webinar demonstrations

Please note that this assignment will involve some amount of self-learning (on the part of modelling solutions appropriately + programming skills)

Submission Deadline: 18th August, 2025

Instructions:

- Read the assignment proposal carefully.
- Solve both assignment problems. Submit two different solution files. (Team # - DQNDDQN, Team # - ActorCritic)
- It is mandatory to **submit** the assignment in **PDF format only** consisting of all the outcomes with each and every iteration. Any other format will not be accepted.
- Add comments and descriptions to every function you are creating or operation you are performing. If not found, then 1 mark will be deducted. There are many assignments that need to be evaluated. By providing the comments and description it will help the evaluator to understand your code quickly and clearly.

How to reach out for any clarifications:

This assignment is administered by

(1) Pooja Harde - pooja.harde@wilp.bits-pilani.ac.in

(2) Divya K - divyak@wilp.bits-pilani.ac.in

(3) Dincy R Arikkat - dincyrarikkat@wilp.bits-pilani.ac.in

Any request for clarification must be addressed through email (official email only) to all the three instructors listed.

Messaging in TEAMS is discouraged. This is to ensure we maintain track of all the transactions. If we find any clarifications to be shared across all the students, we will share this using discussion forums.

-

Part #1 - DQN and DDQN
Total Marks = 7 Marks

Title: Autonomous Drone Battery Management for Urban Surveillance

Scenario: A single autonomous surveillance drone is operating over a simulated urban area. The drone's primary mission is to collect surveillance data by observing points of interest. It has a limited battery and needs to return to one of several charging stations when low on power. The challenge is to manage battery life effectively while maximizing surveillance opportunities, which appear randomly and offer varying rewards. Atmospheric disturbances (wind, rain) also unpredictably affect the drone's energy consumption.

Problem Statement: Develop a Deep Reinforcement Learning agent using Deep Q-Networks (DQN) and Double Deep Q-Networks (DDQN) to optimize the battery usage of an autonomous surveillance drone. The agent must learn an optimal policy to maximize its "surveillance score" by strategically flying, hovering, or returning to a charging station, while adapting to dynamic battery depletion, varying surveillance opportunities (rewards), and random atmospheric disturbances that affect energy consumption.

Key Concepts & State Representation:

- Drone Position: The drone operates on a 2D grid representing the urban area (e.g., a 10×10 grid). Each cell in the grid represents a specific location.
- Battery Level: A continuous value representing the percentage of battery remaining (e.g., 0% to 100%).
- Charging Stations: Fixed locations on the grid where the drone can recharge.
- Surveillance Opportunities (Points of Interest - POIs): These are temporary "hotspots" that appear randomly on the grid. Each POI has:
    - A value (reward) for observing it.
    - A lifespan (how many time steps it remains active before disappearing).
- Atmospheric Disturbance: A continuous value representing the current level of wind/turbulence (e.g., from 0.0 to 1.0, where 1.0 means high disturbance). This directly affects energy consumption.

State Space for input to NN:

- Drone X-coordinate: (Discrete, e.g., 0-9)
- Drone Y-coordinate: (Discrete, e.g., 0-9)
- Current Battery Percentage: (Continuous, e.g., float 0.0 to 100.0)
- Current Atmospheric Disturbance: (Continuous, e.g., float 0.0 to 1.0)
- Information about Active POIs: *(For simplicity, you can limit to just the single "nearest" or "highest value" POI, or use a fixed number of top POIs)*
    - o Proximity to Nearest POI: (Continuous, e.g., Euclidean distance to closest active POI).
    - o Value of Nearest POI: (Continuous, e.g., reward value).
    - o Lifespan of Nearest POI: (Discrete, e.g., remaining timesteps).

Action Space:

- Move North
- Move South
- Move East
- Move West
- Hover: Stay at the current location.
- Recharge: (Only if at a Charging Station). Stay at the station to recharge.

Dynamics and Rewards:

- **Battery Depletion:**
    - Moving: Costs Base_Move_Cost + (Atmospheric_Disturbance * Disturbance_Factor) per step.
    - Hovering: Costs Base_Hover_Cost + (Atmospheric_Disturbance * Disturbance_Factor) per step.
    - Recharging: Gain Recharge_Rate battery percentage per step.
    - Penalty: Large negative reward (-100) if battery reaches 0% and not at a charging station (drone crashes).
- **Surveillance Opportunities (POIs):**
    - Spawning: New POIs appear randomly on empty cells (e.g., 5% chance per timestep).
    - Observation Reward: When the drone is at the *exact location* of an active POI, it collects the POI's value as a positive reward. The POI then disappears.
    - Lifespan Decay: Active POIs decay their lifespan by 1 each timestep; they disappear when lifespan reaches 0.
- **Atmospheric Disturbance**: Changes randomly each time step (e.g., a small random walk, or jump to a new value with some probability).

- **Rewards**:
  - Collecting POI value: +POI_Value.
  - Successful recharge (per step at station): Small positive reward (+1) to incentivize charging behavior.
  - Crash (battery 0% not at station): -100.
  - General time penalty: Small negative reward (-0.1) per step (to encourage efficient operation).

Objective: The agent must learn the optimal policy π∗ using DQN and Double DQN to maximize its cumulative surveillance score (total rewards collected) over many episodes (e.g., each episode is 200-500 time steps).

Environment Setup (Custom Python Simulator):

- A DroneSurveillanceEnv class with __init__, step, and reset methods.
- The step method will handle:
  - Action execution (movement, battery change).
  - Battery depletion based on action and disturbance.
  - POI spawning, lifespan decay, and collection.
  - Atmospheric disturbance updates.
  - Reward calculation.
  - Episode termination conditions (battery crash, max timesteps).
- The reset method will:
  - Place the drone at a starting location.
  - Reset battery to full.
  - Clear active POIs.
  - Set initial atmospheric disturbance.

Requirements and Deliverables: (Total 7 Marks)

1. **Custom Environment Implementation:** Develop the DroneSurveillanceEnv including the __init__, step, and reset methods. Accurately model battery dynamics (depletion, recharge, capacity limits). Implement dynamic POI spawning, value collection, and lifespan decay. Model random atmospheric disturbances and their effect on energy consumption. **(2 Marks)**
2. **DQN Agent Implementation:** Implement a DQN agent capable of interacting with the DroneSurveillanceEnv. **(1 Mark)**
3. **Double DQN Agent Implementation:** Modify the DQN agent's update rule to implement Double DQN and prove how it mitigates overestimation of Q-values in DQN. **(1 Mark)**
4. **Training and Evaluation: (1 Mark)**

    a. Train both the DQN and Double DQN agents over a sufficient number of episodes.
    b. Plot the learning curves (e.g., average cumulative surveillance score per episode over training steps) for both agents.
    c. Compare their performance (average final surveillance score) and training stability. Discuss observations on which algorithm performs better and why.
5. **Policy Analysis: (1 Mark)**
    a. Describe the general learned policy: When does the drone prioritize collecting a low-value, nearby POI versus a high-value, distant one?
    b. How does the agent manage its battery life in relation to POI opportunities and charging station proximity?
    c. Does the agent learn to adapt its behavior (e.g., fly slower, hover more) during high atmospheric disturbances?
    d. Provide examples of an optimal episode's operation (e.g., plot drone path, battery level, POI collections, and disturbances over time).
6. **Hyperparameter Tuning & Discussion: (1 Mark)**
    a. Discuss the impact of key hyperparameters (e.g., learning rate, discount factor, epsilon-greedy exploration strategy, replay buffer size, target network update frequency, neural network architecture) on the training stability and final performance of the agent.

Colab Template: DQN DDQN Code Template (*Make a copy of the template. Do not send the edit access request.*)

PART #2 - Actor Critic
Total Marks = 6 Marks

Title: Actor-Critic Based Sepsis Treatment Optimization

Scenario: You are working as a data scientist in a medical AI research team. Your objective is to train a reinforcement learning agent that can suggest optimal treatment strategies for sepsis patients in the ICU. Sepsis is a life-threatening condition, and its management requires timely interventions based on real-time vitals and clinical decisions. ICU patients are monitored continuously, and treatment decisions (actions) are logged over time.

Objective

The goal of this assignment is to model the ICU treatment process using **Reinforcement Learning**, specifically the **Actor-Critic** method. The agent should learn an optimal

intervention policy from historical ICU data. Each patient's ICU stay is treated as an episode consisting of time-stamped clinical observations and treatments.

Your tasks:

1. Model the ICU treatment process as a Reinforcement Learning (RL) environment.
2. Train an Actor-Critic agent to suggest medical interventions based on the patient's current state (vitals and demographics).

Dataset:

Use the dataset provided in the following link:

https://drive.google.com/file/d/1UPsOhUvyrsrC59ilXsvHwGZhzm7Yk01w/view?usp=sharing

Features:

- **Vitals:** mean_bp, spo2, resp_rate
- **Demographics:** age, gender
- **Action:** Medical intervention (e.g., "Vancomycin", "NaCl 0.9%", or NO_ACTION)
- **Identifiers:** timestamp, subject_id, hadm_id, icustay_id

Environment Setup (RL Formulation)

State Space

Each state vector consists of: *mean_bp (Mean Blood Pressure) , spo2 (Oxygen Saturation), resp_rate (Respiratory Rate), age, One-hot encoded gender*

Action Space

- The agent selects one **discrete action** from 99 possible medical interventions (e.g., Vancomycin, Fentanyl, PO Intake, etc.
- You should **integer encode** or **one-hot encode** these interventions.

Reward

At each time step, the agent receives a **reward based on how close the patient's vitals are to clinically normal ranges**. The reward encourages the agent to take actions that stabilize the patient's vital signs:

$$\text{Reward}_t = -\left((\text{MBP}_t - 90)^2 + (\text{SpO2}_t - 98)^2 + (\text{RR}_t - 16)^2\right)$$

Explanation:

- **MBP** (**mean_bp**): Target = 90 mmHg
- **SpO$_2$** (**spo2**): Target = 98%
- **RR** (**resp_rate**): Target = 16 breaths/min

Each term penalizes the **squared deviation** from the healthy target. The smaller the difference, the higher (less negative) the reward.

Example:

Suppose at time t, the vitals are:

- MBP = 88
- SpO$_2$ = 97
- RR = 20

Then the reward is:

$\text{Reward}_t = -[(88-90)^2+(97-98)^2+(20-16)^2] = -(4+1+16) = -21$

A lower (more negative) reward indicates worse vitals, guiding the agent to learn actions that minimize this penalty.

<u>Episode termination</u>

An episode ends when the ICU stay ends. To define this:

1. **Group the data by subject_id, hadm_id, icustay_id**
   → Each group represents one ICU stay = one episode.

2. **Sort each group by timestamp**
   → Ensure the time progression is correct.

3. **For each time step in a group (i.e., each row). Check if it is the last row in that group.**
   → If yes, then mark done = True (end of episode).
   → If no, then done = False (continue episode).

Requirements and Deliverables:(Total 6 Marks)

1. Load the dataset containing ICU patient records, vitals, demographics, and actions. **(0.5 Mark)**
   a) Convert timestamps to datetime format and sort by time within each ICU stay.
   b) Encode categorical columns such as gender and action.
2. Develop the SepsisTreatmentEnv Environment. **(0.5 Mark)**

   a) Define state as a vector containing: mean_bp, spo2, resp_rate, age, gender
   b) Define action space as a list of 99 unique discrete treatment actions, including 'NO_ACTION'.
   c) Define the episode structure: One ICU stay = One episode
3. Implement the Reward Function based on patient vitals **(1 Mark)**
4. Design and Train Actor-Critic Algorithm using appropriate neural network architectures **(2.5 Mark)**
5. Plot the graph for Average episode reward over training and Episode lengths **(1 Mark)**
6. Provide a 200-word writeup on the behavior, reward trends, and stability of the trained policy **(0.5 Mark)**

Colab Template: Actor Critic Code Template(*Make a copy of the template. Do not send the edit access request.*)