Problem Statement

You are hired as a Sr. Machine Learning Engineer at Super Cabs, a leading app-based cab provider in a large Indian metro city. In this highly competitive industry**,** retention of good cab drivers is a crucial business driver, and you believe that a sound **RL-based system for assisting cab drivers** can potentially retain and attract new cab drivers.

Cab drivers, like most people, are incentivised by a healthy growth in income. The goal of your project is to build an RL-based algorithm which can help cab drivers maximise their profits by improving their decision-making process on the field.

The Need for Choosing the "Right" Requests

Most drivers get a healthy number of ride requests from customers throughout the day. But with the recent hikes in electricity prices (all cabs are electric), many drivers complain that although their revenues are gradually increasing, their profits are almost flat. Thus, it is important that drivers choose the 'right' rides, i.e. choose the rides which are likely to **maximise the total profit** earned by the driver that day.

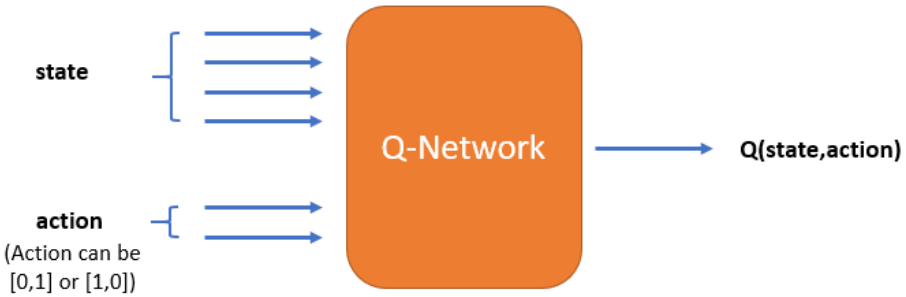
For example, say a driver gets three ride requests at 5 PM. The first one is a long-distance ride guaranteeing high fare, but it will take him to a location which is unlikely to get him another ride for the next few hours. The second one ends in a better location, but it requires him to take a slight detour to pick the customer up, adding to fuel costs. Perhaps the best choice is to choose the third one, which although is medium-distance, it will likely get him another ride subsequently and avoid most of the traffic.

There are some basic rules governing the ride-allocation system. If the cab is already in use, then the driver won’t get any requests. Otherwise, he may get multiple request(s). He can either decide to take any one of these requests or can go ‘offline’, i.e., not accept any request at all.

Markov Decision Process

Taking long-term profit as the goal, you propose a method based on reinforcement learning to optimize taxi driving strategies for profit maximization. This optimisation problem is formulated as a Markov Decision Process. The major simplifying assumptions of the MDP is listed in the attached file MDP.pdf.

In this project, you need to create the environment and an RL agent that learns to choose the best request. You need to train your agent using **Deep Q-learning (DQN)** . You are free to choose any of the two architectures of DQN listed below.



Architecture 1

Architecture 2

You can **optionally refer** the paper *Deep Reinforcement Learning for List-wise Recommendations* by Xiangyu Zhao, Liang Zhang, Zhuoye Ding. The paper has mentioned a few recommendations on how to select the Q-network architecture.

It's up to you to choose the Q-network architecture. You will find the following files along with the project.

- Environment file - Env.py
- Agent Architecture 1 - DQN_Agent_Arch1.ipynb
- Agent Architecture 2 - DQN_Agent_Arch2.ipynb
- Time-Matrix - TM.npy

Goals

1. **Create the environment:** You are given the 'Env.py' file with the basic code structure. This is the "environment class" - each method (function) of the class has a specific purpose. Please read the comments around each method carefully to understand what it is designed to do. Using this framework is not compulsory, you can create your own framework and functions as well.
2. **Build an agent that learns to pick the best request using DQN**. You can choose the hyperparameters (epsilon (decay rate), learning-rate, discount factor etc.) of your choice.
    0. Training depends purely on the epsilon-function you choose. If the ε decays fast, it won't let your model explore much and the Q-values will converge early but to suboptimal values. If ε decays slowly, your model will converge slowly. It is recommended that you try converging the Q-values in 4-6 hrs.
    1. In the Agent file, code skeleton is provided. Using this structure is not necessary though.

Submission Requirements:

1. Submit your code along with the final DQN model as a pickle file
2. Prepare a report with convergence plots.