Final Project Report-WiDS

Analysis of Geopolitical factors over Global Supply Chain using RL (Mentor: Adarsh Prajapti)

Siddhant Singhania
24B3956

1. Basics of Reinforcement Learning

Reinforcement Learning (RL) is an ML technique where an **agent** interacts with an **environment** by taking **actions** and receiving **rewards** to get adapted to a policy with best suited returns. It is my job to design these actions, states and rewards to get the best results.

The basic components are:

- **State (s):** Different positions an agent can be in
- **Action (a):** What the agent can do
- **Reward (r):** The feedback
- **Policy (π):** States to action map
- **Value / Q-value:** Expected reward in the future

Over the course of this project, I dived from the basics of RL to understanding its use-cases in inventory management.

2. What I Did in Assignment 1

Problem Overview

In Assignment 1, the task was to solve the **FrozenLake environment** using:

- **Q-Learning (off-policy)**
- **SARSA (on-policy)**

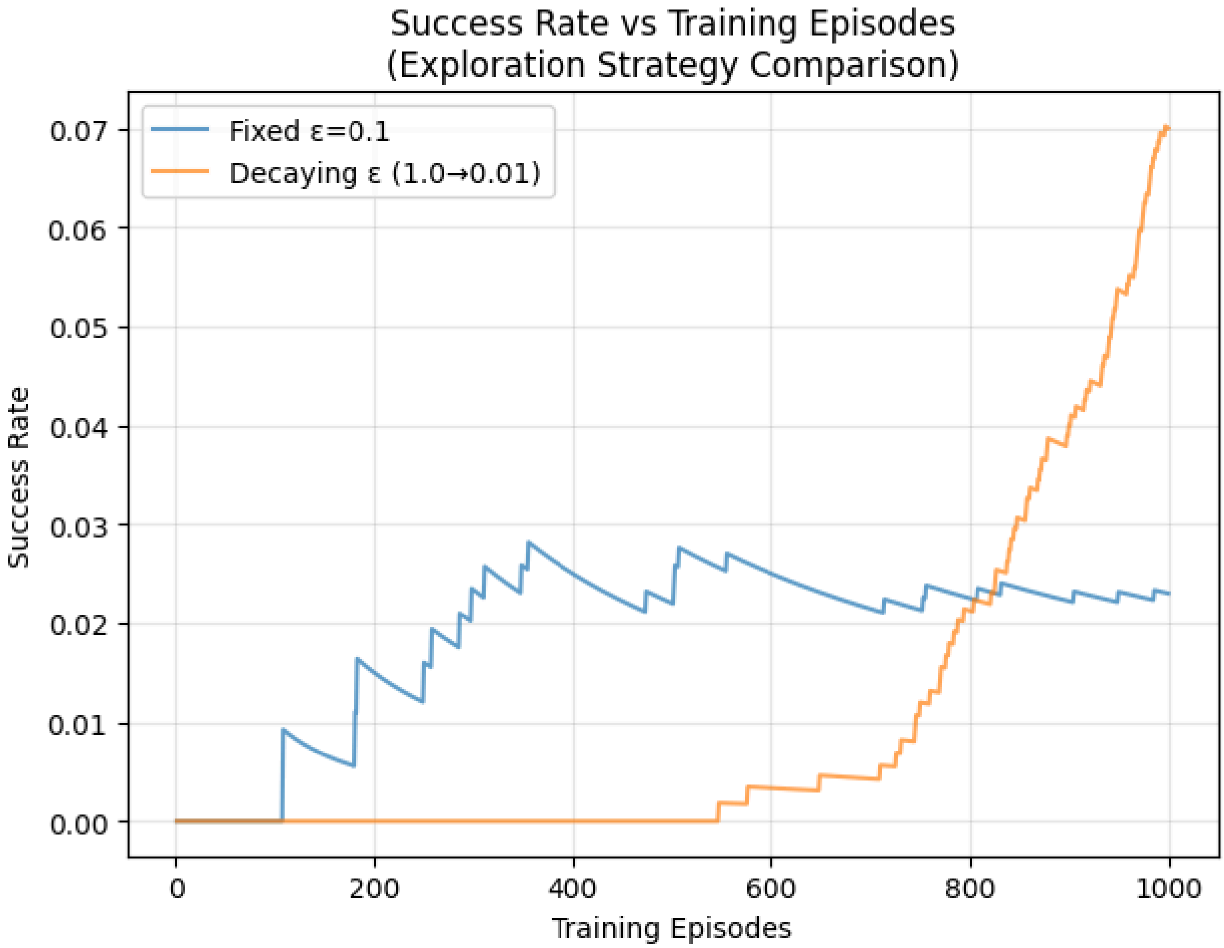
The goal was not just to reach the target, but to understand:

- How tuning epsilon-greedy exploration rates affect learning
- How hyperparameters change behavior

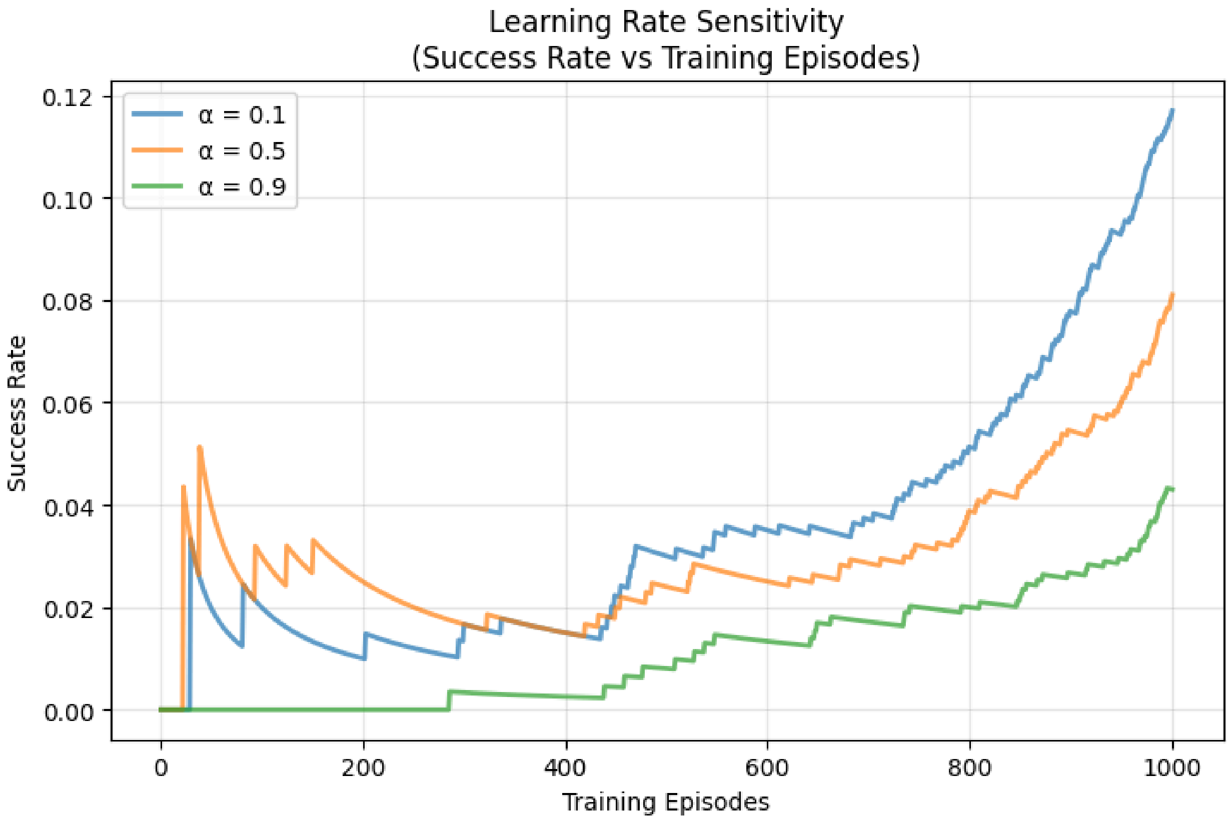
Key experiments:

- Fixed ε vs decaying ε
- Different learning rates (α)
- Different discount factors (γ)

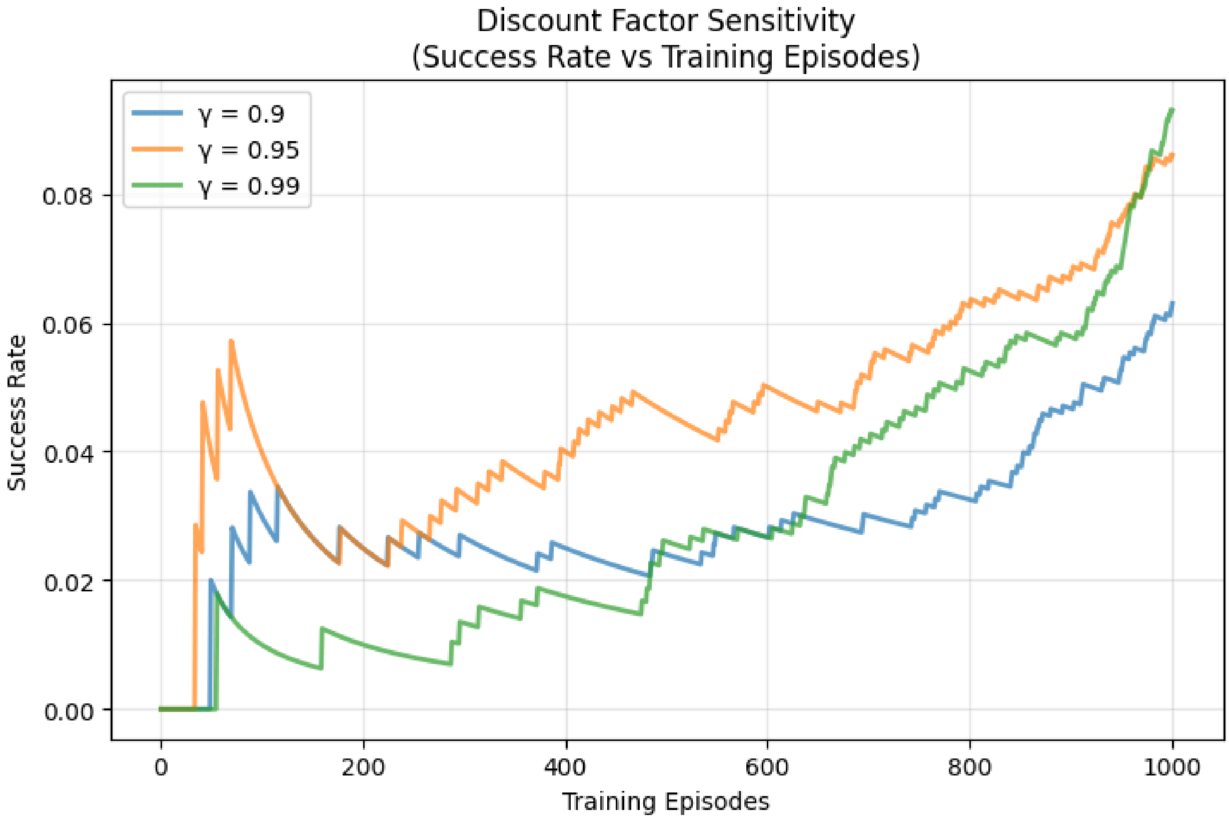
Observations:



1) Decaying ε shows a better success rate, and that too which is directly proportional to the number of training episodes after a point. This is because it technically dynamically solves the exploration-exploitation problem by constantly changing the parameter value, exploring new strategies to obtain more optimal paths.



2) There is a band getting created where the less the value of the learning rate, more is the success rate. In the end it is maybe characteristic to this problem, but a lower learning rate prevents overfitting and loss of good policies for suboptimal policies, so eventually with a large number of training episodes this saturates out to such a band. This was an interesting observation for me too.



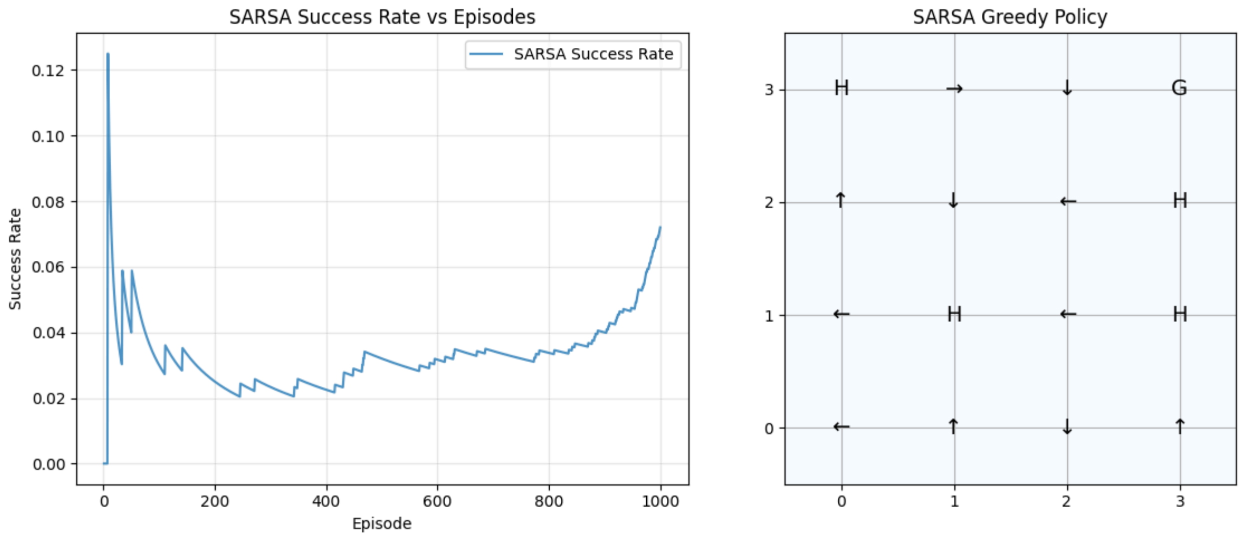
3) Discount factor shows an opposite trend - the bands created are such that the higher the discount factor, more is the success rate after a large number of training episodes. The possible justification for this is that a larger discount factor means that the agent considers long term rewards. We know that the discount factor gets multiplied in a GP like fashion across future rewards - the reward after the third action would have a coefficient of γ^2, the reward after the fourth action would have that of γ^3, and so on. So a large discount factor prevents us from neglecting future rewards, which is good for a long term run when we consider a large number of training episodes.

SARSA

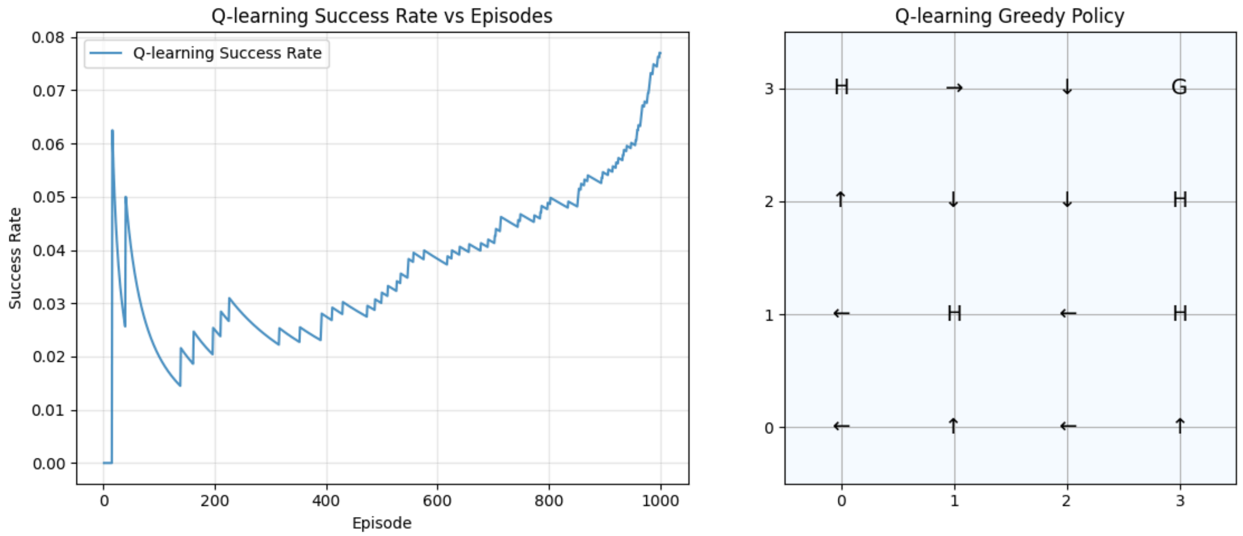
- Implemented **on-policy SARSA**

The difference is that Q learning makes a decision based on the *next optimal action*, which is why in the Q value calculation we have a max() term to signify this. In SARSA, it just takes the next action rather than the next optimal action, meaning it doesn't try to create an optimal decision from the start. It is more of a high-risk high reward system then, and it helps this algorithm recognise actual risky paths which have a very low Q value again due to this linear exploration, making it technically **more robust.**

Observation:
We can see a comparison of Q-Learning and SARSA over here.



3	H	→	↓	G
2	↑	↓	←	H
1	←	H	←	H
0	←	↑	↓	↑
	0	1	2	3



3	H	→	↓	G
2	↑	↓	↓	H
1	←	H	←	H
0	←	↑	←	↑
	0	1	2	3

As you can see, SARSA shows a higher result peak and a more rising graph, showing better success rate after number of episodes. It is less evident, but these results show some truth in the theory. Again, this might be because of bad parameter setting since we carried in the same parameter, or maybe judging a higher number of episodes for our conclusions should have been a more sound technique.

Conclusions

- **Off-policy vs On-Policy theoretically** have a major difference, but in our observations we weren't able to see much. Maybe they are visible for a more complex model.

- Exploration nature affects success rate, along with hyperparameter tuning. We have to choose the correct parameters based on what we prioritise - short term or long term learning. We also have to factor in computing speed.

---

3. Assignment 2: Driving on a Circular Track

Problem Overview

In Assignment 2, the task was to:

- Design a **continuous control environment, using physics** to run a car on an annular track in the most optimum way by setting initial position, velocity and other conditions and then using RL and self created states, actions, and a reward system to achieve this result. The problem here was the reward system, it is not simple unlike traditional RL problems.

4. What I Did in Assignment 2

State Space

I designed a space vector as a state, which consisted of the following parameters:

- **x, y** → Car position in 2D
- **v** → Linear velocity
- **θ** → Inclination of car from reference
- **ω** → Angular velocity

Action Space

- **a** → Acceleration / deceleration
- **δ** → Steering angle (−90° to +90°)

This was the action space, and a few of these were already instructed in the assignment. Steering angle for example, was a suggestion.

Reward Space

This was the ultimate challenge, because after a detailed discussion with my mentor I realised it's not a simple -1 or 1 based reward.

- **Angular Progress Reward**

The agent is rewarded for moving forward along the circular track, meaning it is progressing towards completing a full lap.

- **Time Penalty**

A constant negative reward is applied at every time step, to prevent SLOW behaviour. Because it is also our objective to complete a lap in the fasted time

- **Radial Deviation Penalty**

_A penalty proportional to the distance of the car from the center of the track is applied, such that it increases near the edges of the annular track and forces the car to be in the center.

- **Speed Incentive**

More speed, less time, more reward.

- **Out-of-Track Penalty**

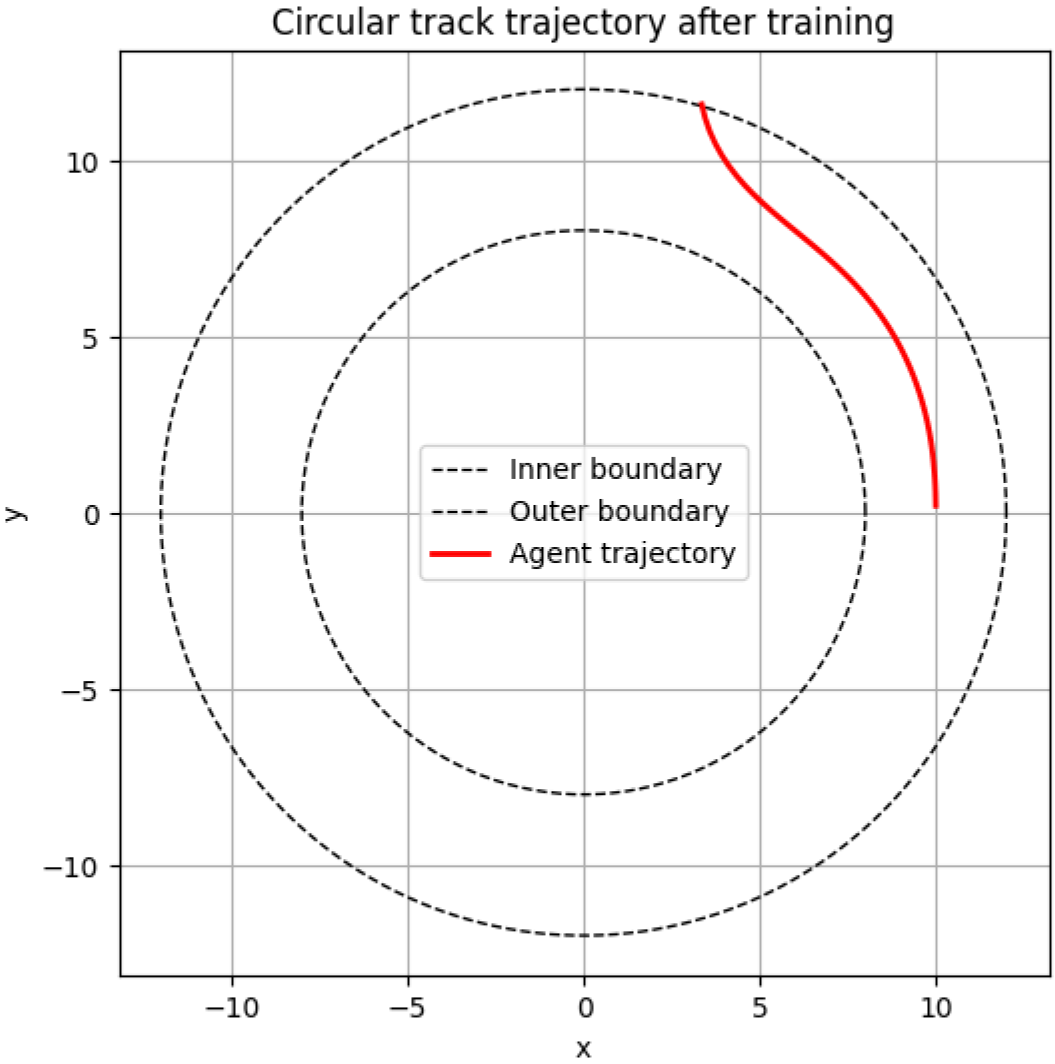
A large negative reward is applied if the vehicle leaves the region.

Why PPO?

Our entire action space is continuous. We are dealing with acceleration and steering angles, there is no such **set** of actions, there is a range of different real values these parameters can take. If we use traditional Q-Learning, it would involve creating an action space of 1000s, maybe 10s of 1000s of different possible actions and even rewards.

PPO uses a neural network, it is designed for gradual changes in policy. It is able to handle tasks well which are genuinely continuous in nature because of this added advantage. It is also on-policy, meaning it is a more robust policy option as seen in assignment 1.

Observations



We can see how the learning process of the agent brought it to a near perfect radial path at the center of the annular track.

Observed statistics showed that **MEAN EPISODE LENGTH** was terminating early initially due to early stages of training, where track exit happened quicker. Majorly negative reward also suggested that initial penalties dominated. **POLICY GRADIENT LOSS** having a low value suggested gradual policy development. However, eventually these statistics delved down to stable values suggesting completion of the model in an appropriate manner.

Conclusions

- **Successfully implemented** my first complex RL Model of continuous states and actions using PPO and self designed reward system.

---

5. MADRL Supply Chain Paper - the link between RL and Supply Chains

My mentor gave me this paper - **Multi-Agent Deep Reinforcement Learning for Multi Echelon Inventory Management**, which deals with applying Deep-Learning based MARL in the realm of IEOR, in particular inventory management.

- Design a multi-agent system with multiple actors from different ranks and organisations involved. We treat these as autonomous agents who work together to achieve certain goals.
- Every actor learns a policy by resorting to **all the other actors' observations, but making decisions only based on their own observations in execution.**
- This is just a MARL problem extended to solve every actor's inventory management. The paper strictly specifies that this policy of information sharing between different actors or agents only happens in the training stage to make it implementable in **practical supply chain settings**.
- Actors from echelons included
    - Manufacturer
    - Distributor
    - Warehouse
    - Retailer
    - Customer
- They together form a supply chain network system.
- The dynamics behind this were that the ith actor received orders from i-1 and replenished inventory from i+1. Other important terms were the customer demand, actor backlogs, holding costs, quantity of goods being shipped. These terms were then varied with different statistical distributions such as Poisson and also varied with real life data and used to construct apt MARL policies using deep-learning.

Learnings

- Understanding of the MARL dynamics in comparison to simple RL.
- The parameters, terms, states involved when extending this problem to inventory management.
- Understanding the use of different statistical distributions and relations to use to fit and train models and their impact on model accuracy.
- Exploring new and different modelling techniques and comparing their results to obtain newer patterns - the paper displayed how different effects showed different variations in costs and demand fluctuations when modelled by such effects.

---

6. Overall Learnings

- Built strong fundamentals in RL.
- Built a mental aptitude for solving complex problems in RL by developing reward and action spaces. The car problem was one such problem as it helped me literally develop the black box algorithm to solve it.
- Understanding complex MARL algorithms and their applications in solving rich problems and the scale of this problem. It really intrigued me as to what all I can further learn in the realm of RL.