Traffic Signal Control Using Reinforcement Learning Agents

Presented by:
Abhijit Taneja
Anjali Ramchandani
Shan Sun
Tanmay Patil

UNIVERSITY OF CALIFORNIA, RIVERSIDE

Problem

- Traffic congestion.
- Bad traffic flow could lead to:
  - Fuel emissions
  - Delays, which can further have cascading effects

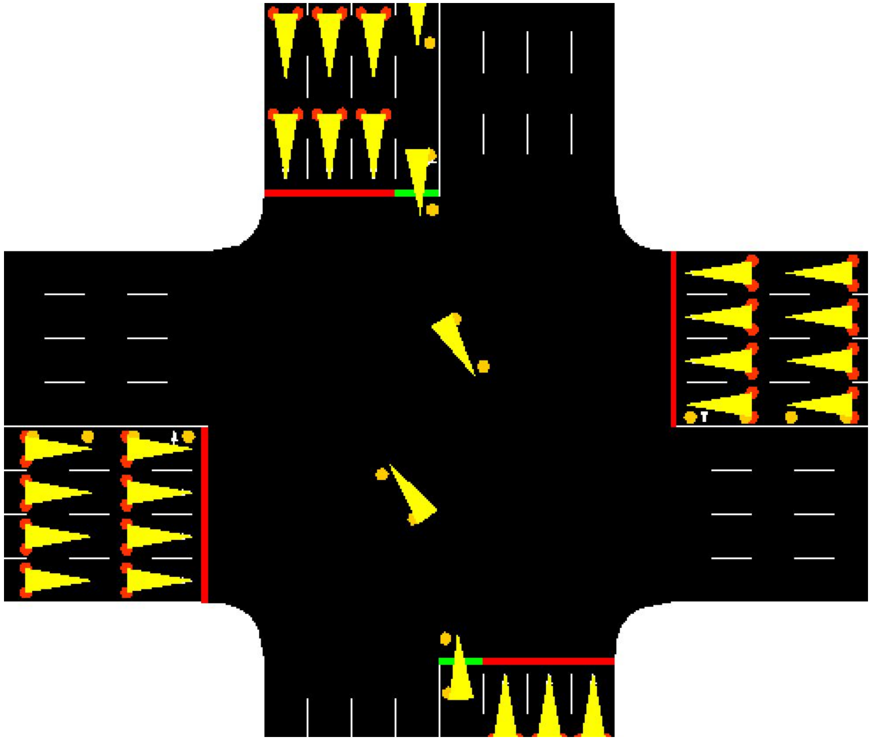
Approach

Adaptive Traffic Signal

- **There are various models existing that use reinforcement learning**
- **Reinforcement learning algorithms can vary between value-based methods or actor-critic methods.**
- **We have compared the results from Q-learning (value-based) and A2C (actor-critic) methods**

Simulation

- **Environment**
  A 4-way intersection with each arm having 2-way 4-lane roads. We will be considering only the possibility of a left turn



- **Dataset**
  1000 cars generated over 5400 frames. Traffic signals with green duration 10 seconds and yellow 4 secs. The cars follows the behaviour of a ideal driver.

SUMO

- **SUMO**
  "Simulation of Urban MObility" (Eclipse SUMO) is an open source, highly portable, microscopic and continuous road traffic simulation

- **NETEDIT**
  A command line tool which helps map routes and network xml files into SUMO configuration file

- **TRACI**
  API for SUMO library. Useful for changing the flow of events

Learning Agents

Learning Parameters

- **States**
  We divided the positions of car with respect to traffic lights into 80 states.

- **Actions**
  Change in traffic signal to green or yellow is considered one unique action. 4 ways so 4 action.

- **Rewards**
  Cumulative sum of weighting time of all the cars is taken to be the reward. We only take the negative part in order to penalize the model.

- **Results**
  We chose to decide the efficiency of the model based total of cars halted per step, until the last step

Deep Q Learning

UCR

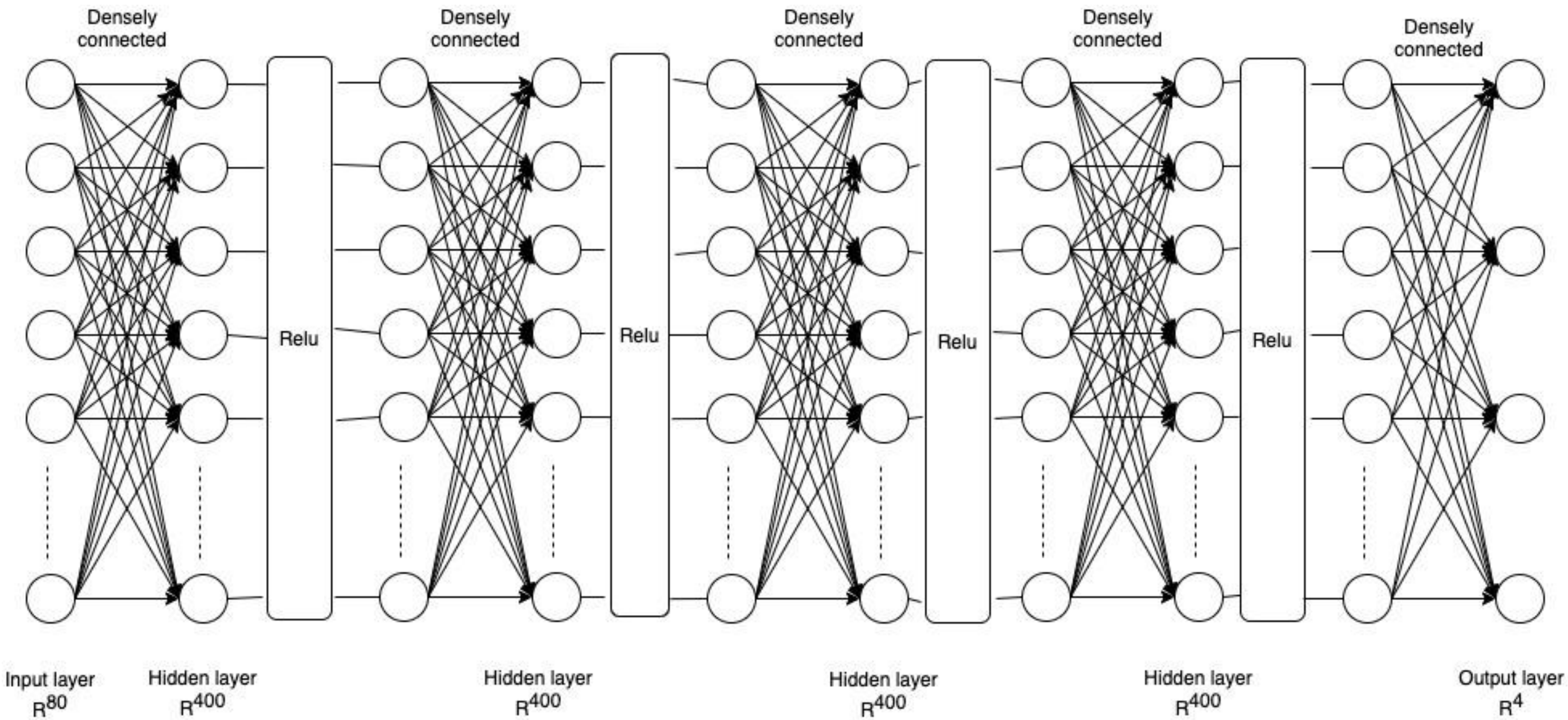
- We apply Q Learning algorithm to into order to train you model for the best state action pair

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

- In deep Q-learning, we use a neural network to approximate the Q-value function. The state is given as the input and the Q-value of all possible actions is generated as the output.

```
for i, b in enumerate(batch):
    state, action, reward, nextState = b[0], b[1], b[2], b[3]  # extract data from one
    qCurr = qs[i]
    qCurr[action] += alpha * (reward + self._gamma * np.amax(qns[i]) - qCurr[action])
    x[i] = state
    y[i] = qCurr
```

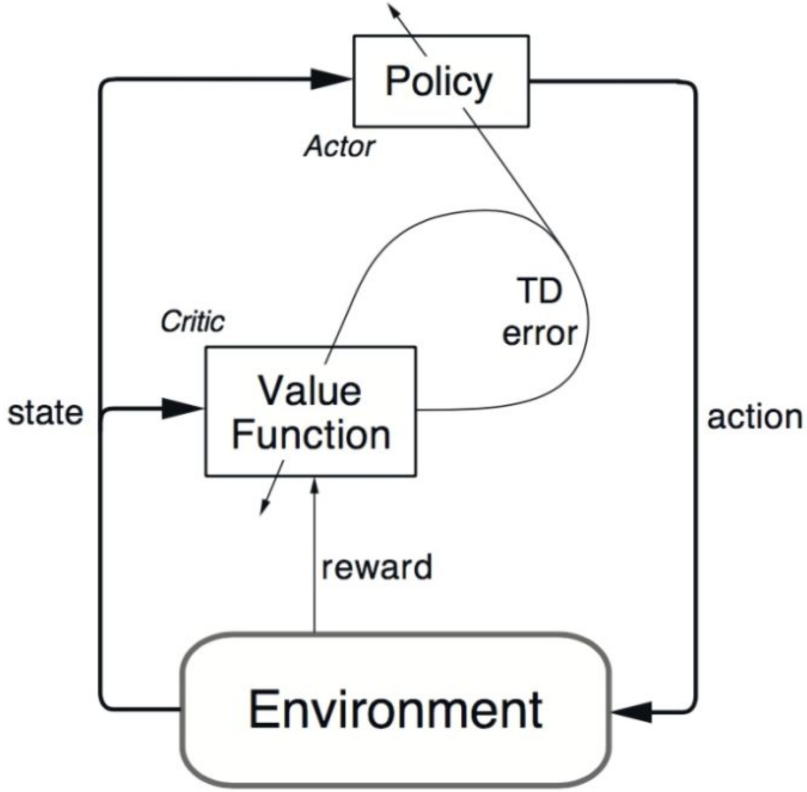
DQL NN Architecture



Inputs: current states s_t
Output: action-value function $Q(s_t, a_t)$

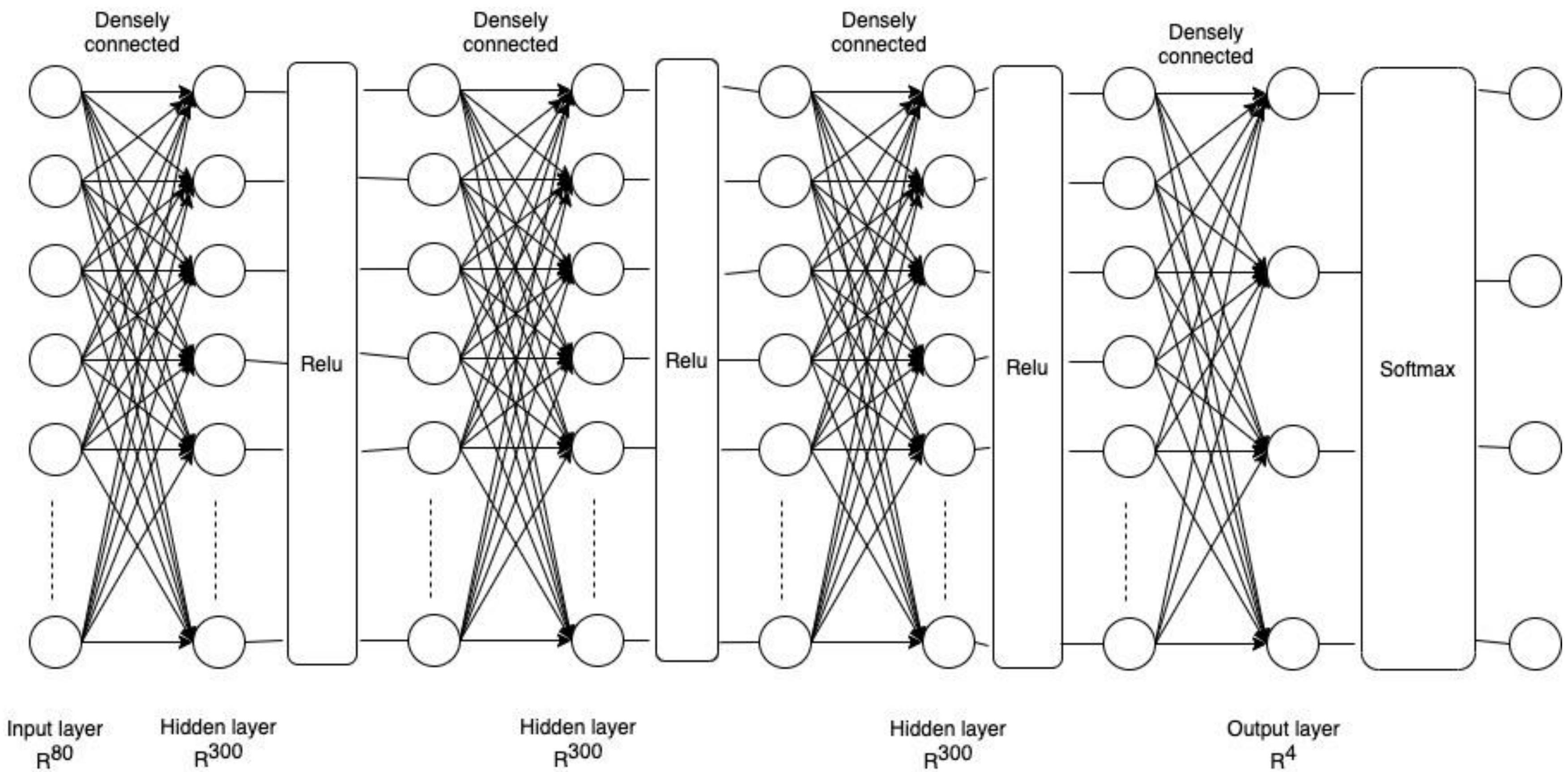
A2C

- A2C decomposes the Q (state-action) value into two pieces: the state Value function V(s) and the advantage value A(s, a).



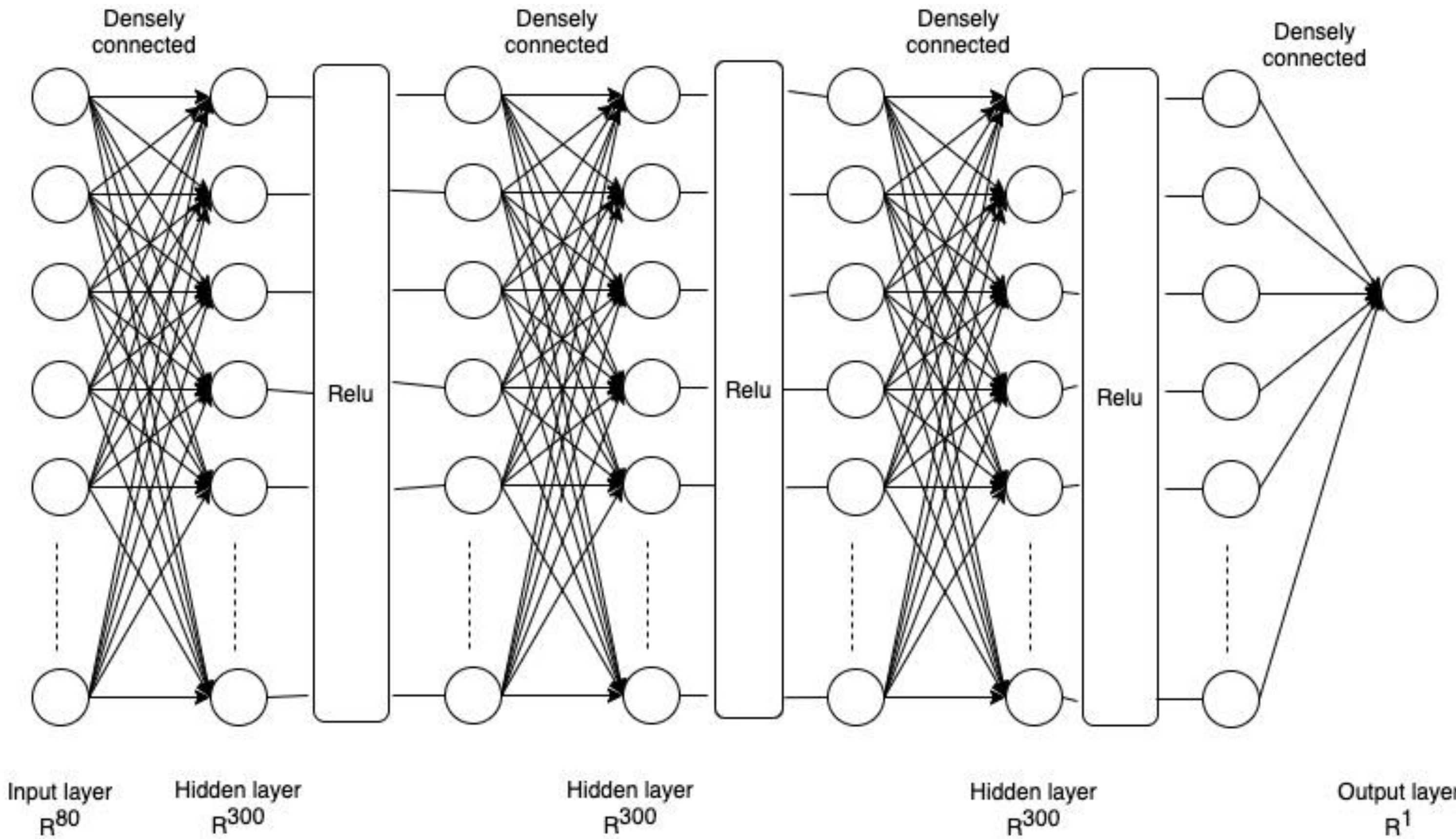
- Actor decides which action to take and critic tells the actor how good its action was

A2C Actor Architecture



Inputs: current states s_t
Output: policy $\pi(a_t|s_t)$

A2C Critic Architecture

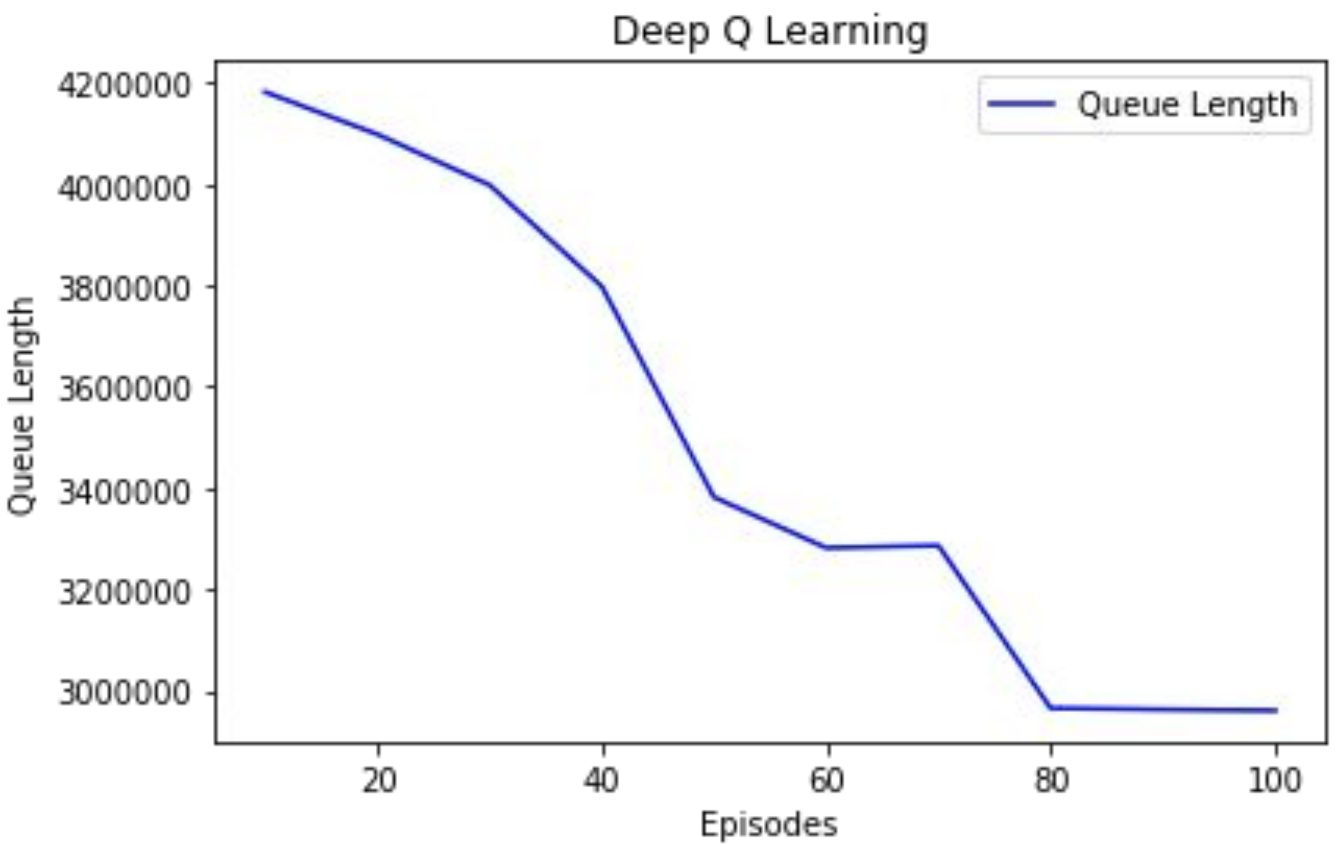


Inputs: current states s_t
Output: value function $V(s_t)$

Results and analysis

Deep Q learning Network

- DQL Show consistent drop in queue length. There is a hint of saturation not can't be sure.



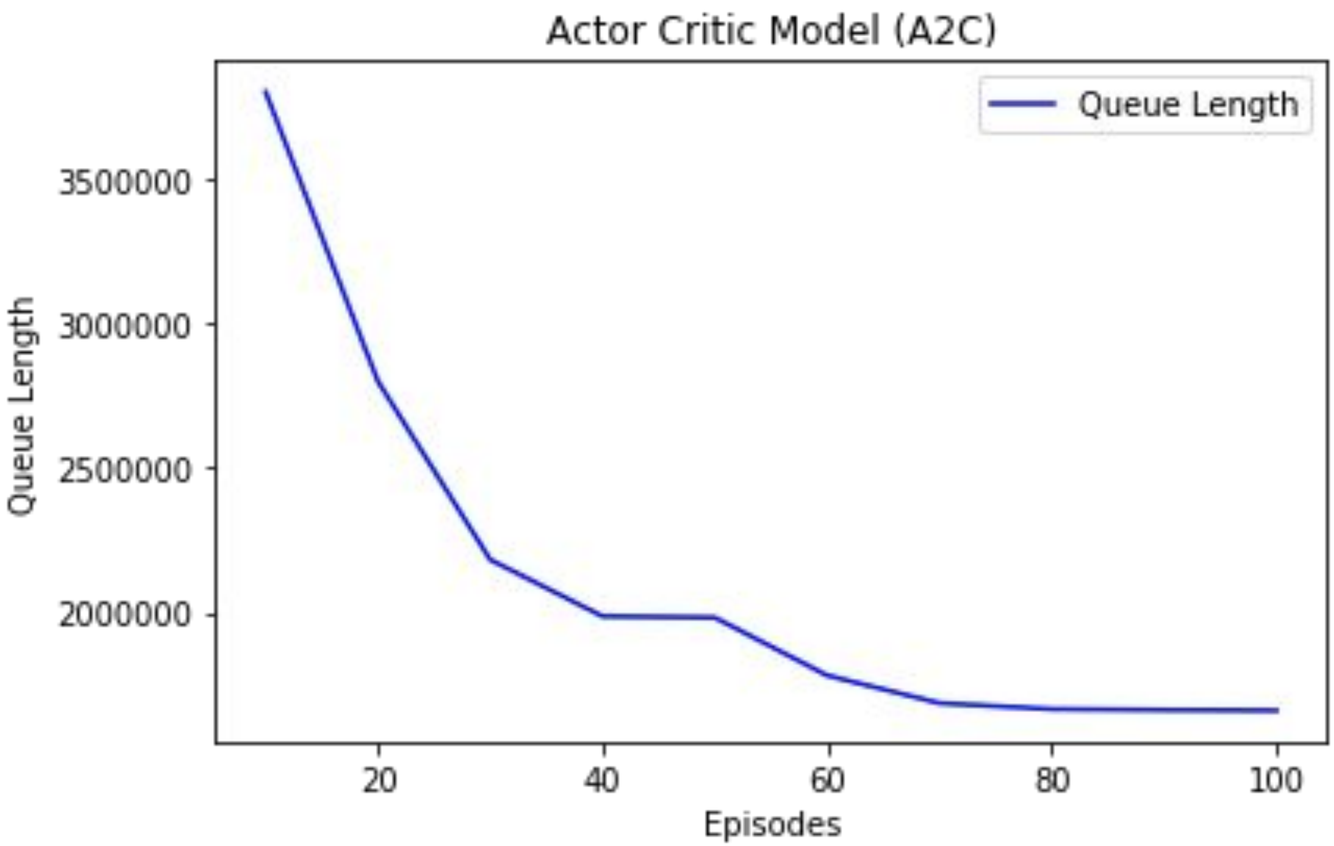
Deep Q learning Network

```
Begin Testing

----- Test episode

Loading configuration ... done.

Total reward: -62100

Queue Length Sum: 2981374

Simulation time: 266.0 s

 Testing done
```

Output for 100 episodes with 100 epochs

Actor Critic Network

- The A2C model shows heavy drop till 40 episodes, after that it starts to saturate



Actor Critic Network

```
Begin Testing

----- Test episode
 Retrying in 1 seconds
Loading configuration ... done.

Total reward: -29346.0
Queue Length Sum: 165904
Simulation time: 9.6 s

Testing done
```

Output for 100 episodes with 100 epochs

Thank You!
Questions?