COL778: Principles of Autonomous Systems
Assignment 2

Harshit Goyal (2021MT10143)

March 2025

Please find the project repository here.

Contents

1 Tabular Q-Learning

1.1 Learning Policy

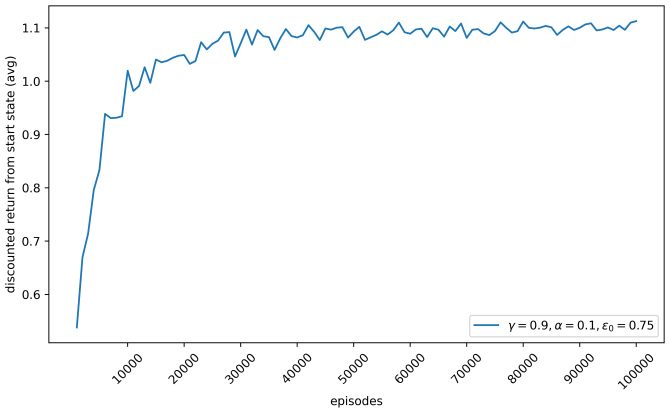

Figure 1: Discounted Return

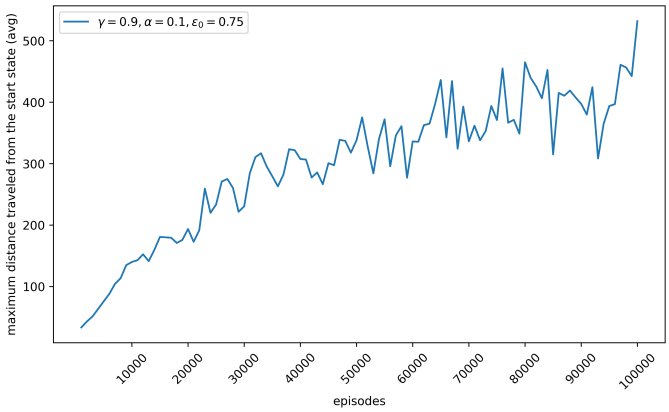

Figure 2: Distance Traveled

Episodes	Discounted Return	Average Distance
50,000	1.09	338.00
98,000	1.10	456.64
99,000	1.11	442.55
100,000	1.11	531.98

Figure 3: `Tabular` $\gamma = 0.9, \alpha = 0.1, \epsilon = 0.75$

1.2 Lane Visualization

1. See Figure 4. `Black` to `white` represents increasing Q-value. Note that colors in two different visualizations are not to the same scale.
2. The agent assigns highest value to the lane where it can avoid collision for the longest time.
3. The agent also seems to value a lane based on value if lane(s) adjacent to it.

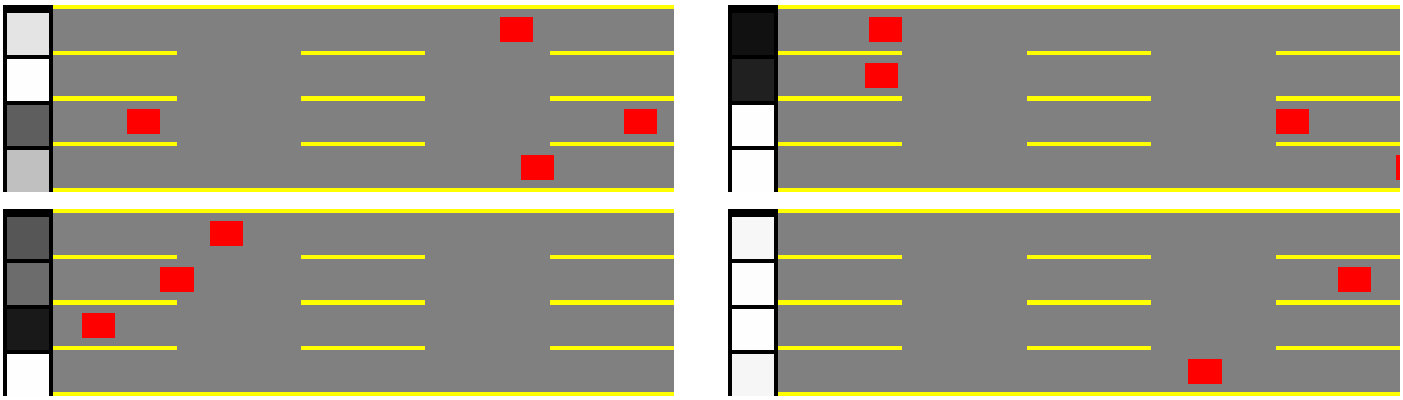

Figure 4: Lane visualization for Tabular Q-agent

1.3 Speed Visualization

1. Note that `speed = 4` is always black due to the environment.
2. See Figure 5. The agent prefers higher speed when the obstacle is farther away.

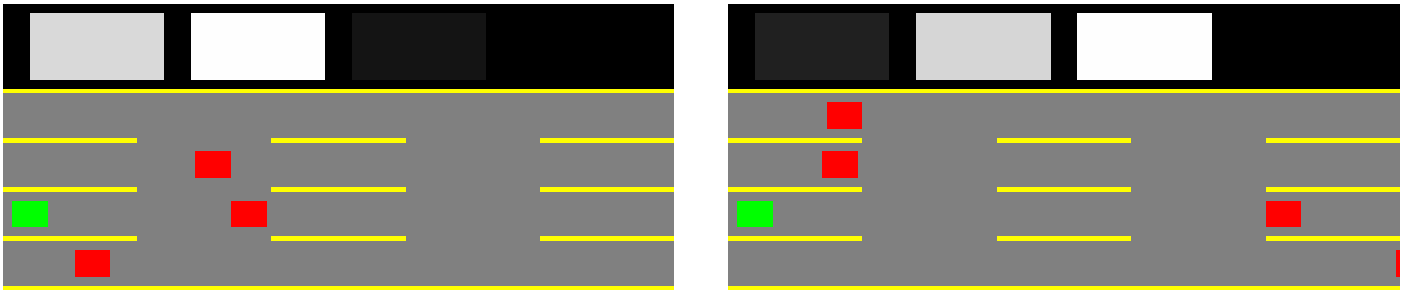

Figure 5: Speed visualization for Tabular Q-agent

1.4 Varying γ

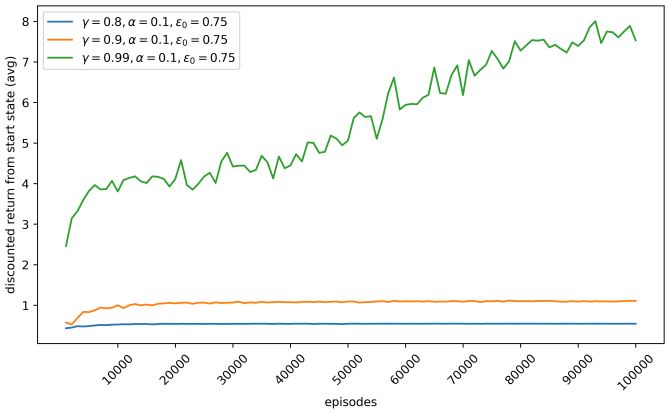

Figure 6: Discounted Return

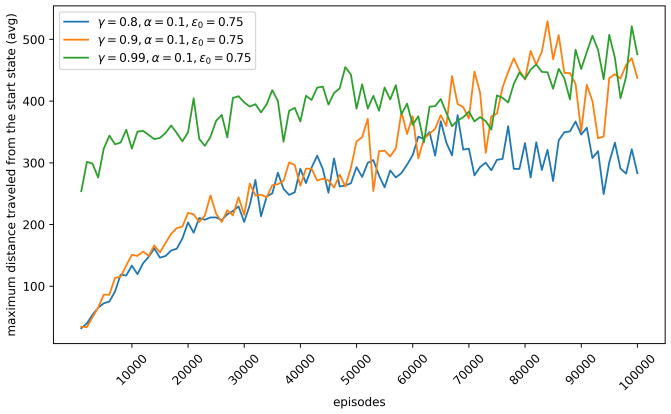

Figure 7: Distance Traveled

γ	Discounted Return	Average Distance
0.80	0.55	283.21
0.90	1.11	437.49
0.99	7.53	475.81

Figure 8: `Tabular` $100,000$ iterations, $\alpha = 0.1, \epsilon = 0.75$

Increase in the discount factor tends to lead to a significant increase in the discounted returns, (as upper bound on discounted returns is $\frac{0.12}{1-\gamma}$) and a general increase in the average distance traveled. The reason is discussed in Reward-Goal Relationship.

1.5 Varying α

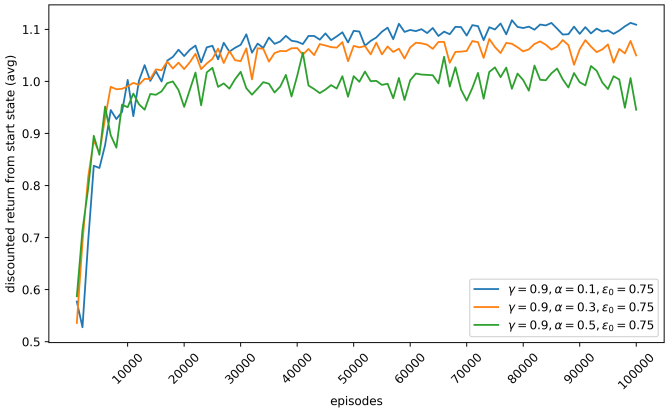

Figure 9: Discounted Return

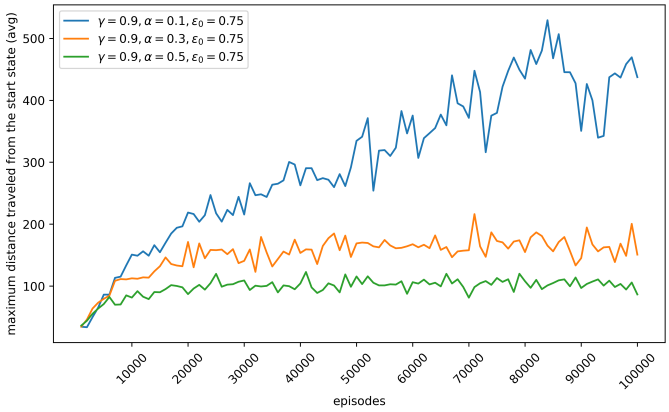

Figure 10: Distance Traveled

α	Discounted Return	Average Distance
0.1	1.11	437.49
0.3	1.05	150.90
0.5	0.95	86.61

Figure 11: `Tabular` $100,000$ iterations, $\gamma = 0.9, \epsilon = 0.75$

In `Tabular` training, an update is made after every action. The batch size is 1. Higher α means bigger updates leading to unstable training, and agent being biased towards later updates. Very briefly, during the initial few thousand episodes, higher α does better. Later, "forgetting" starts.

1.6 Scheduling ϵ

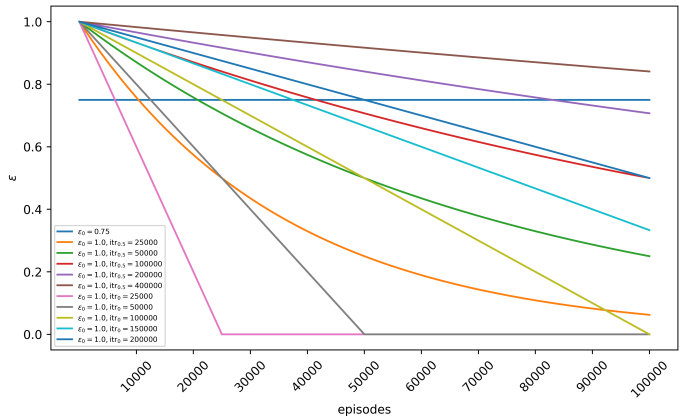

Figure 12: ϵ while training

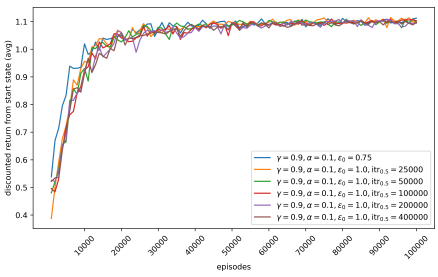

Figure 13: Discounted Return

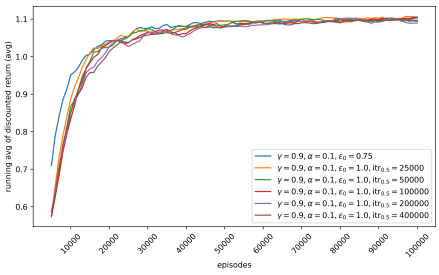

Figure 14: Running Average of Discounted Return

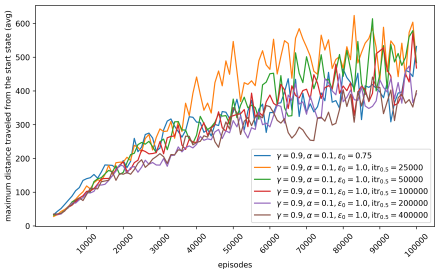

Figure 15: Distance Traveled

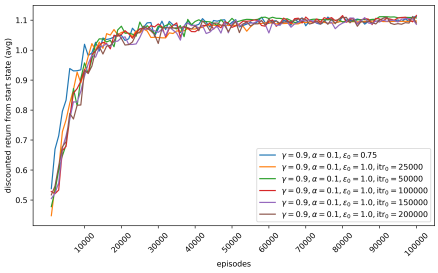

Figure 16: Discounted Return

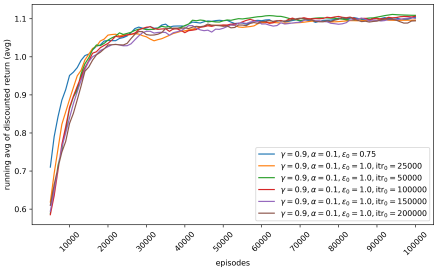

Figure 17: Running Average of Discounted Return

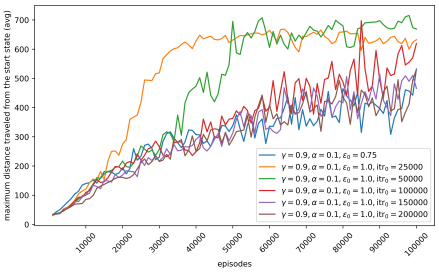

Figure 18: Distance Traveled

$\texttt{itr}_0$	Discounted Return	Average Distance
$25,000$	1.097	632.68
$50,000$	1.108	668.74
$100,000$	1.116	619.39
$150,000$	1.086	466.01
$200,000$	1.091	531.35

Figure 19: Linear Scheduling ϵ

$\texttt{itr}_{1/2}$	Discounted Return	Average Distance
$25,000$	1.097	483.09
$50,000$	1.099	484.38
$100,000$	1.103	467.94
$200,000$	1.094	391.59
$400,000$	1.096	400.47

Figure 20: Exponential Scheduling ϵ

Figure 21: `Tabular` $100,000$ iterations, $\gamma = 0.9, \alpha = 0.1, \epsilon_0 = 1.0$

In Linear Scheduling, Average Distance and Discounted Reward) peak at $\texttt{itr}_0$ $50,000$ and $100,000$ respectively. Too fast ϵ decay leads to starvation and too slow decay does not give the model enough opportunity to lean the optimum policy, given the knowledge of rewards explored.

2 Changing Environment

2.1 Overtake Reward

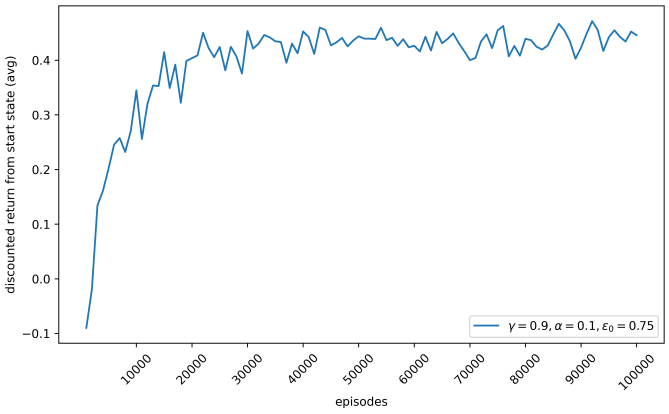

Figure 22: Discounted Return

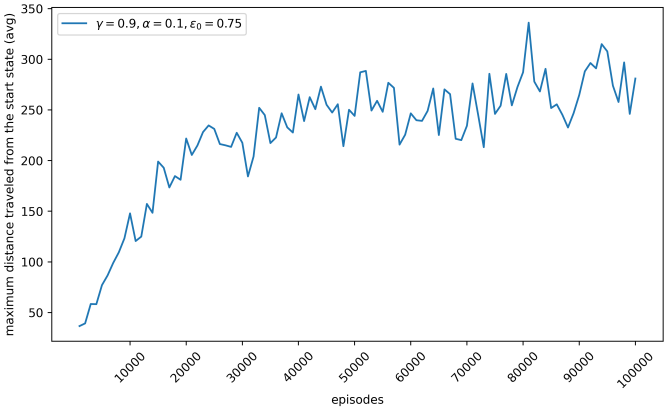

Figure 23: Distance Traveled

Episodes	Discounted Return	Average Distance
98,000	0.43	296.87
99,000	0.45	246.07
100,000	0.45	280.99

Figure 24: `Tabular` $\gamma = 0.9, \alpha = 0.1, \epsilon = 0.75$. Reward: `overtakes`.

1. From the `gifs` we observed that the agent drives faster (trying to overtake the other cars). This is in contrast to the case then reward is total distance traveled, where the agent tries to move in lanes that are empty and also reduces it's speed often in order to **remain far** from other car, to reduce chances of collisions with other cars.
2. Lane Visualization: See Figure 25. We didn't see any clear difference from the total distance reward case. The agent prefers lanes where obstacles are farther away.
3. Speed Visualization: See Figure 26. The agent has a clear preference here for higher speeds, even when it is close to other cars, the agent doesn't assign higher value to slowing down, unlike when reward is total distance.

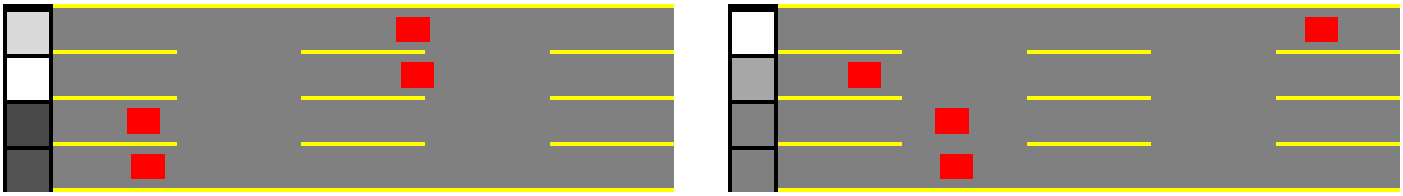

Figure 25: Lane Visualization for Tabular Q-agent with `overtakes` reward

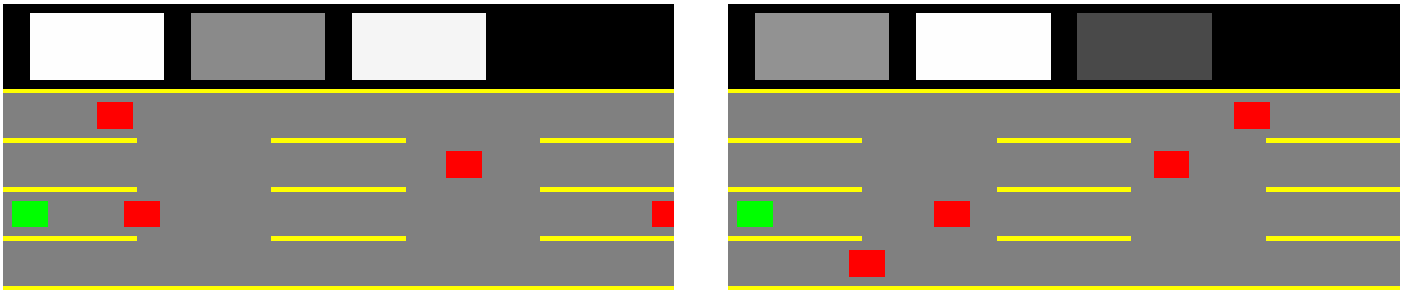

Figure 26: Speed Visualization for Tabular Q-agent with `overtakes` reward

2.2 Distance levels: 3

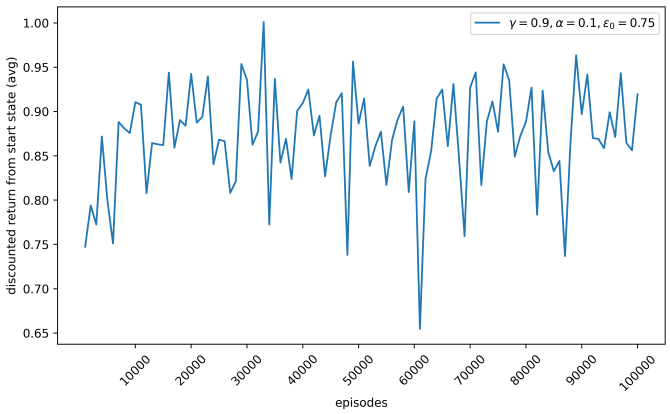

Figure 27: Discounted Return

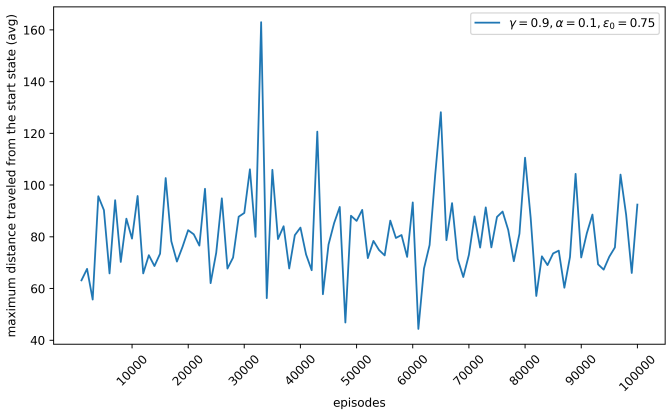

Figure 28: Distance Traveled

Episodes	Discounted Return	Average Distance
98,000	0.86	88.52
99,000	0.86	66.02
100,000	0.92	92.46

Figure 29: `Tabular` $\gamma = 0.9, \alpha = 0.1, \epsilon = 0.75$ and 3 discrete levels of distance.

1. We see that the total distance traveled is much less compared o the case of 5 discrete levels. In the `gifs` we see that the agent starts "acting" (switching lanes) even when it is much far from the cars (compared to the 5 levels case).
2. Lane Visualization: See Figure 30. The agent is not able to differentiate well due to smaller level of discretization.
3. Speed Visualization: See Figure 31. Compared to Figure 5, the difference is assigned value is lower for the similar settings. The obstacle has to come really close to the agent for it to notice.

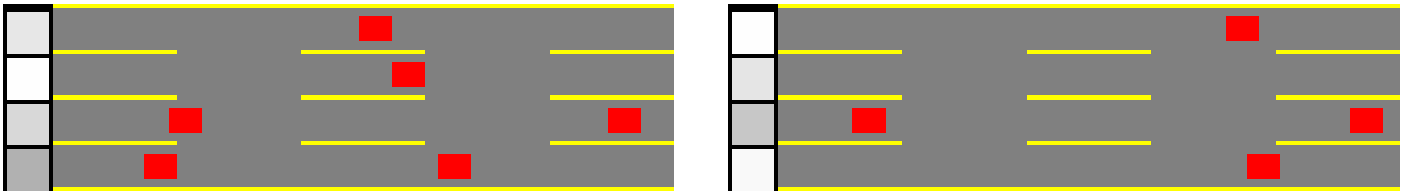

Figure 30: Lane Visualization for Tabular Q-agent with 3 levels of distance discretization

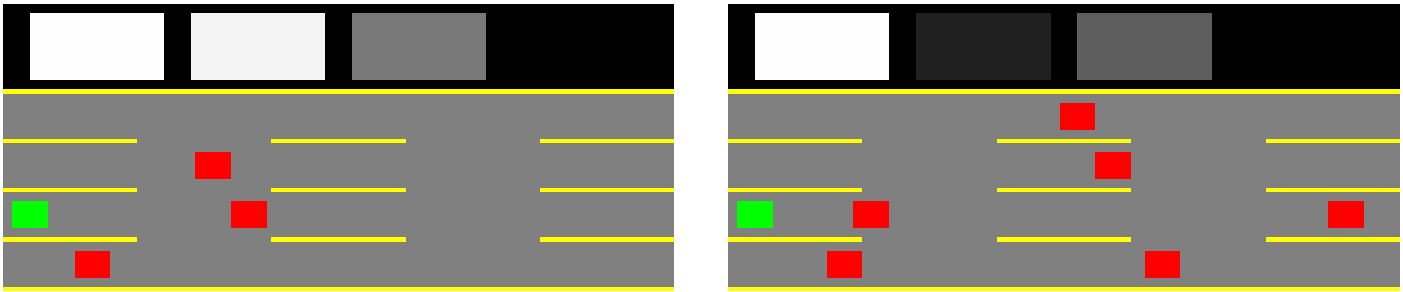

Figure 31: Speed Visualization for Tabular Q-agent with 3 levels of distance discretization

3 Deep Q-Learning (Neural Network as Q-Function)

3.1 Architecture

1. For the Q-function, we use a Neural Network network with 2 hidden layer of 32 units each. The output layer has 5 outputs , one corresponding to each action.
2. For the `discrete` case there are total $\mathtt{n_{speed}} \times \mathtt{n_{lanes}} \times \mathtt{n_{dist}^{n_{lanes}}} = 10,000$ states and we use embeddings of size 16.

3. For the `discrete` case there are total $\texttt{n}_{\texttt{speed}} \times \texttt{n}_{\texttt{lanes}} = 16$ states and we use embeddings of size 4. The distances d_1, d_2, d_3, d_4 are directly concatenated.

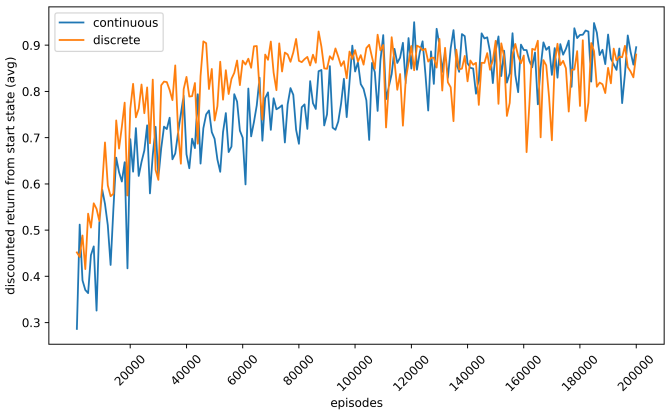

Figure 32: Discounted Return

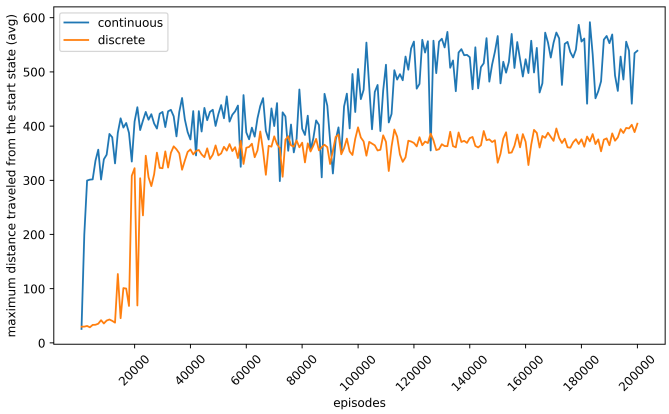

Figure 33: Distance Traveled

Model	Episodes	Discounted return	Average Distance
`Tabular`	$100,000$	1.11	531.98
`DQN discrete`	$200,000$	0.88	404.42
`DQN continuous`	$200,000$	0.90	538.91

Figure 34: $\gamma = 0.9$, `Tabular` trained with $\alpha = 0.1, \epsilon = 0.75$, `DQN` trained with `Adam` optimizer, learning rate $= 10^{-4}$, batch size 256 and replay buffer size $10,000$. For reference, maximum possible total discounted return is 1.2.

`DQN continuous` does better than `DQN discrete`. We believe this is because the d_1, d_2, d_3, d_4 are fed directly into the model. So the ordering, that for values of $d_1 = 0.2, 0.6, 0.9$ the order of closeness is implied. While in the discrete case, this has to be learned by the embedding layer. `DQN continuous` acts "identically" on $d_1 = 0.49$ or $d_1 = 0.51$.

3.2 Visualization: `DQN Discrete`

See Figure 35 and Figure 36. The values assigned by the agent are absurd, also reflected in the low performance of the agent in Figure 34.

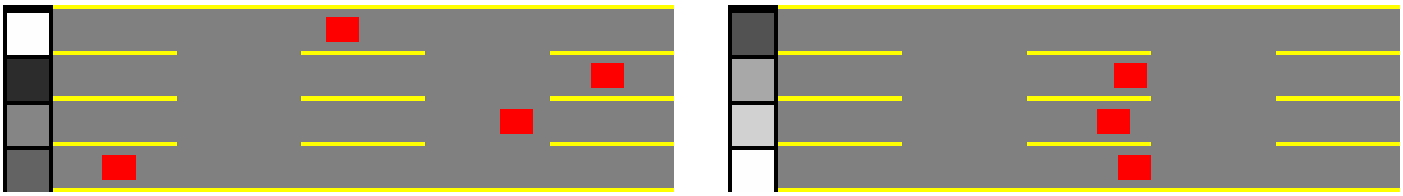

Figure 35: Lane Visualization for `DQN Discrete` agent

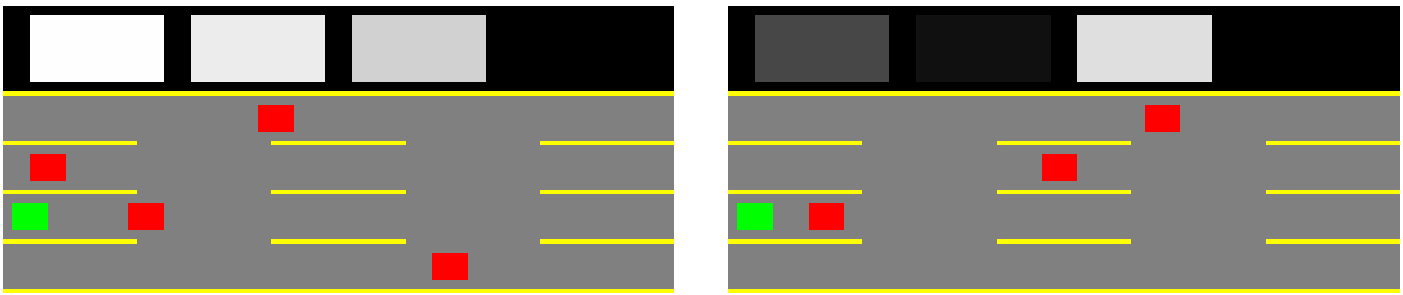

Figure 36: Speed Visualization for `DQN Discrete` agent

3.3 Visualization: `DQN Continuous`

See Figure 37 and Figure 38. Preferences assigned are explainable, like in the Tabular-Q agent.

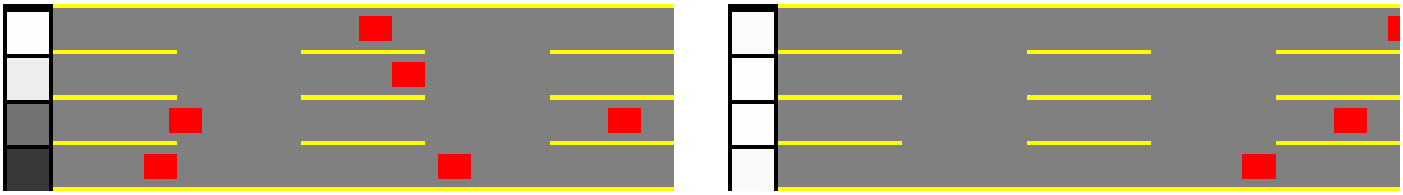

Figure 37: Lane Visualization for `DQN Continuous` agent

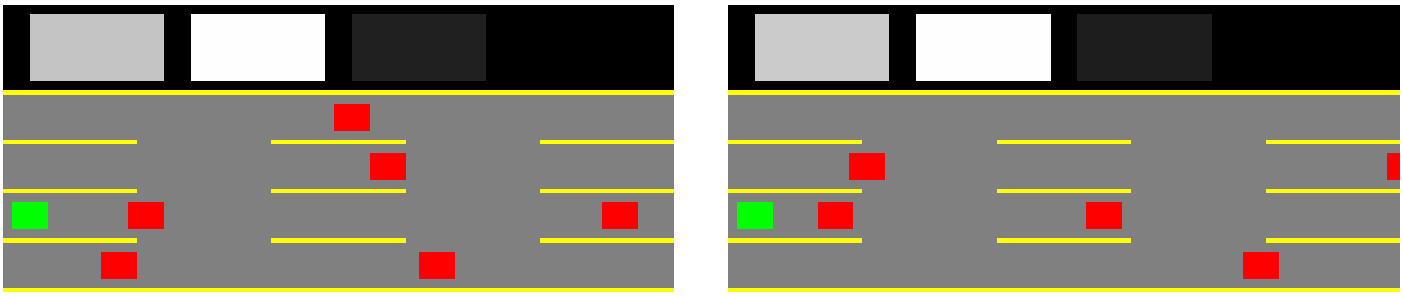

Figure 38: Speed Visualization for `DQN Continuous` agent

4 Experimenting and Improving the Agent

4.1 Reward-Goal Relationship

Out goal is to maximize the total distance traveled by the agent. The total discounted return (that the agent tries to maximize) is a proxy for the goal.

$$r(state_t, a_t, state_{t+1}) = \begin{cases} -5 & \texttt{if control car collides} \\ 0.03 * s_{t+1} & \texttt{otherwise} \end{cases} \tag{1}$$

$$\texttt{Distance} = \Delta t(s_1 + s_2 + s_3 \dots)$$
$$\texttt{Discounted Return} = 0.03(s_1 + \gamma s_2 + \gamma^2 s_3 \dots)$$

1. While training, we see that sometimes the total discounted return earned increases while total distance traveled decreases. This might happen because the car prioritizes higher speeds earlier and that may lead to a crash.
2. As we increases $\gamma \to 1$, the total discounted return is more aligned with the total distance but training becomes unstable, often also reducing performance.

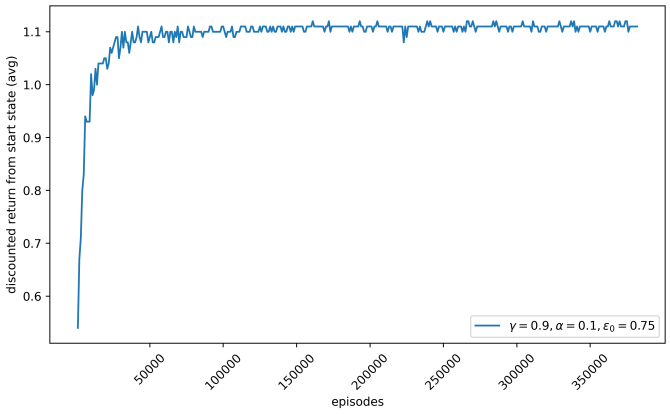

Figure 39: Discounted Return

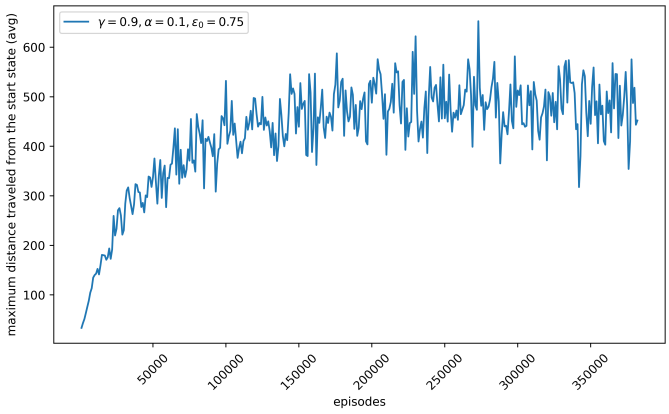

Figure 40: Distance Traveled

Figure 41: `Tabular` $\gamma = 0.9, \alpha = 0.1, \epsilon = 0.75$. Further training after $100,000$ iterations does not improve performance.

4.2 Double DQN

Here, a separate network is used for computing `target`. This network is updated "slowly". This approach aims to stabilize training. In vanilla DQN, the `prediction` and `target` networks both keep moving.

$$\texttt{target} \leftarrow r + \gamma \max_{a'} Q_{target}(s_{t+1}, a')$$
$$\texttt{prediction} \leftarrow Q_{policy}(s_t, a_t)$$

Q_{policy} is trained using gradients. Q_{target} is updated as

$$\theta_{target} \leftarrow (1-\tau)\theta_{target} + \tau\theta_{policy}$$

4.3 Experiments: Optimizers

1. We experimented with `Adam` and `AdamW` optimizers with various variations of `amsgrad`, `weight_decay` and γ intending that more stable training would allow increasing γ. The NN architecture used was same as in Architecture.
2. We also trained with batch size 512 but it didn't help. We then moved to a deeper network for the Q-function, described in Experiments: `DDQN` leading to the `DDQN-Vanilla` model.
3. While training `DDQN-Vanilla` and `DDQN-Time-Embed` we also clip gradients to 100.

4.4 Designing Reward

1. Results in Figure 19 suggest that linearly decreasing ϵ works much better than other strategies. In the `DDQN` case we saw however that the distance traveled down starts going down later. We set a minimum on the ϵ.
2. To penalize earlier crashes more, we set the reward for crash to

$$\texttt{reward}_{\texttt{crash}} = -5\left(1 - \frac{\texttt{num\_steps}}{\texttt{episode\_max\_length}}\right)$$

3. To favor longer runs we add an increasing living reward

$$\texttt{reward}_{\texttt{living}} = 0.15\frac{\texttt{num\_steps}}{\texttt{episode\_max\_length}}$$

4. For the above reward to work, we also add the current timestep as $\left(\frac{\texttt{num\_steps}}{\texttt{episode\_max\_length}}\right)$ to the state.

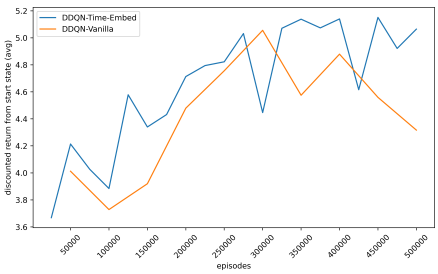

Figure 42: Discounted Return

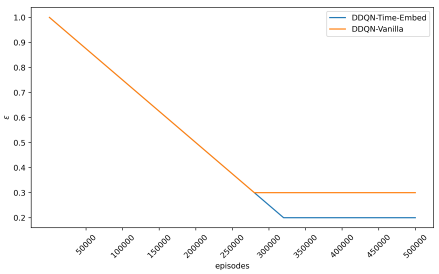

Figure 43: ϵ during training

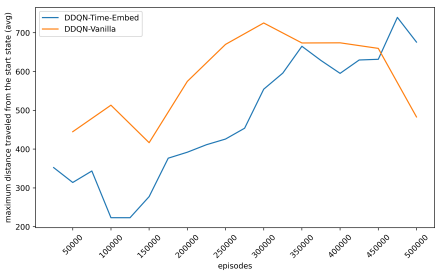

Figure 44: Distance Traveled

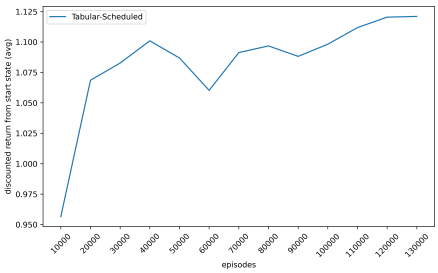

Figure 45: Discounted Return

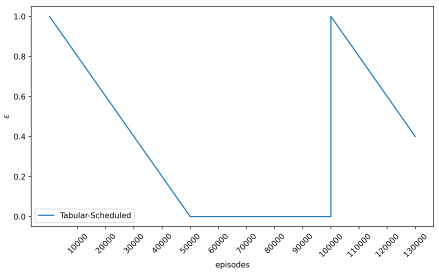

Figure 46: ϵ during training

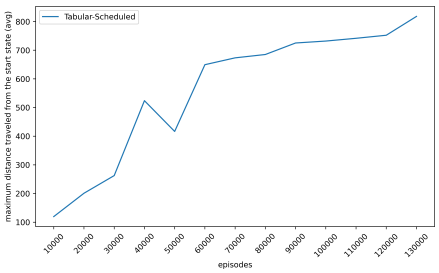

Figure 47: Distance Traveled

Episodes	Discounted Return	Distance Traveled
400,000	5.14	595.23
475,000	4.92	739.43
500,000	5.06	675.82

Figure 48: `DDQN-Time-Embed`

Episodes	Discounted Return	Distance Traveled
300,000	5.06	725.08
400,000	4.88	673.96
500,000	4.32	482.94

Figure 49: `DDQN-Vanilla`

Episodes	Discounted Return	Distance Traveled
50,000	1.09	416.63
100,000	1.10	731.93
130,000	1.12	817.92

Figure 50: `Tabular-Scheduled`

4.5 Experiments: `DDQN`

1. Both `DDQN` trained with $\gamma = 0.98, \alpha = 0.1$. `continuous` observations. The architecture used has the same embedding described in Architecture but

$$64 \texttt{ (input layer)} \to 128 \to 128 \to 64 \to 5 \texttt{ (action layer)}$$

   units respectively in the Neural Network. `AdamW(lr = 0.0001, amsgrad = True)` is used for training with batch size 512 and $\tau = 0.005$.

2. `DDQN-Time-Embed` Reward described in Designing Reward and ϵ decreased linearly from 1.0 to 0.2 in 320,000 iterations then kept constant.

3. `DDQN-Vanilla` Reward same as original, Equation 1 and ϵ decreased linearly from 1.0 to 0.3 in 280,000 iterations then kept constant.

4. Neither of these models would fit in the 2-hours time limit, and the training time also make it difficult to discover hyper-parameters.

4.6 `BestAgent: Tabular-Scheduled`

1. We trained the `Tabular` agent for 100,000 episodes with $\gamma = 0.90, \alpha = 0.1$ and ϵ decreased linearly from 1.0 to 0.0 in 50,000 episodes. After 100,000 episodes we train it with $\gamma = 0.90, \alpha = 0.01$ for 30,000 episodes. ϵ follows the same schedule as before.

2. Over 100,000 seeds from 0 to 99,999, the `Tabular-Scheduled` policy gave an average distance 812.11 with standard deviation 276.30.

5 Viva Feedback

1. The `discrete DQN` implementation is not ideal. The speed levels and distance levels are discretized but have an inherent ordering. Using an embedding layer introduces entanglement into otherwise ordered and separable states, increasing the data requirement for effective learning.

2. This is reflected in the lower performance of `DQN discrete` and slower initial learning (see Figure 33).

3. A naive strategy is to keep driving at speed 1. Other cars have speed 1 or 2. Over a total of 1000 episodes, the distance traveled is $0.3 \times 1000 \times 1 = 300$, where 0.3 is the time per step. The agent does not need to switch any lanes and will not collide.