

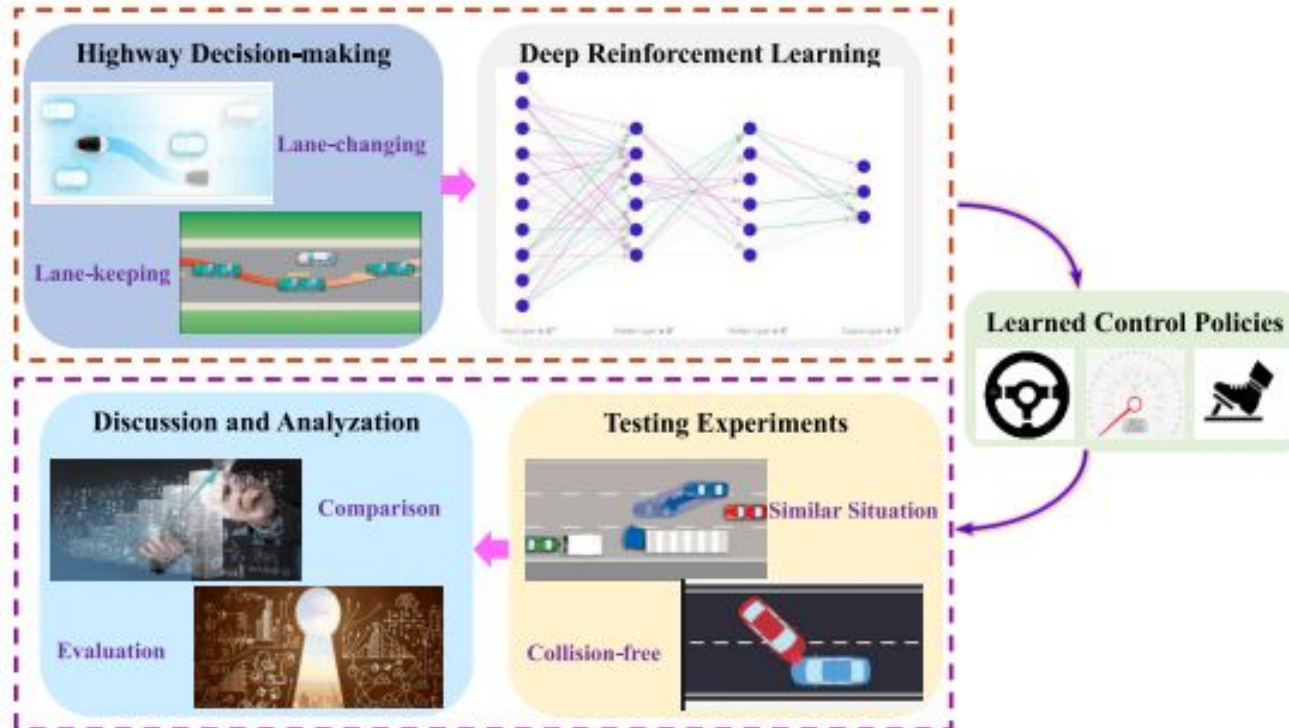
**Team #8 Final Project**

# **Decision-Making Strategy on Highway for Autonomous Vehicles Using Deep Reinforcement Learning**

<b>Name</b>	<b>Email</b>	<b>ASU ID</b>
<b>Siddahant Jain</b>	<b><u><a href="mailto:sjain198@asu.edu">sjain198@asu.edu</a></u></b>	<b>1223151020</b>
<b>Dhiram Ashok Buch</b>	<b><u><a href="mailto:dbuch@asu.edu">dbuch@asu.edu</a></u></b>	<b>1221194052</b>
<b>Arshnoor singh Sachdeva</b>	<b><u><a href="mailto:asachd11@asu.edu">asachd11@asu.edu</a></u></b>	<b>1222482300</b>
<b>Monark Bandishbhai Parekh</b>	<b><u><a href="mailto:mparekh2@asu.edu">mparekh2@asu.edu</a></u></b>	<b>1222179426</b>

# Abstract

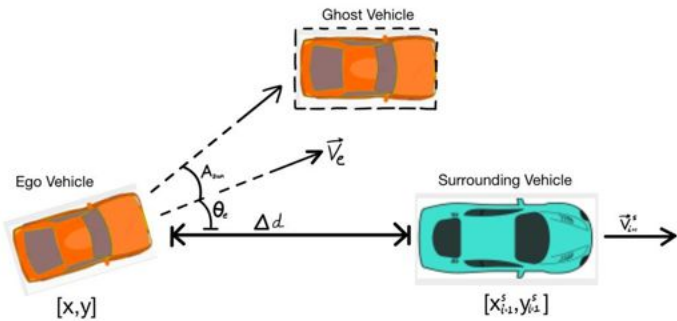
Autonomous driving is a promising technology that has the potential to improve the driving efficiency and reduce accidents. In this project, we explore the **overtaking maneuvers of self-driving vehicles in a highway environment** using deep reinforcement learning techniques: **Deep Q-Network (DQN)** and **Dueling-Deep Q-Network (DDQN)**. Firstly, we prepare a highway driving environment where in the ego vehicle attempts to overtake surrounding vehicles using safe maneuvers. **The ego vehicle has two level of controls: upper level and lower level.** Upper-level control is responsible for making highway decision making, like when to overtake or when to slow down, etc., and the lower-level control is responsible for inputs like steering angle, throttle input etc. Then, we implement the reinforcement learning techniques to the upper-level controller for formulating a policy and evaluate the advantages and shortcomings of different implementations.



Overview of the Project

# Introduction: Model Definitions

**State space:** Lateral and yaw dynamics of a vehicle are the state space of the model in hierarchical control structure



$$s_{set} = \{s_{ego}, s_{sur}\}$$

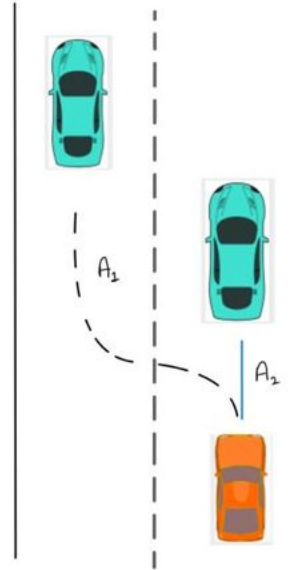
$$s_{ego} = \{x, y, v, a_e, \theta_e, \Delta d, t_{ttc}, A_{steer}, A_{throttle}, A_{brake}\}$$

$$s_{sur} = \{v_i^s, \Delta v_i^s, \Delta x_i^s, \Delta y_i^s\}_{i=1,2,...,n}$$

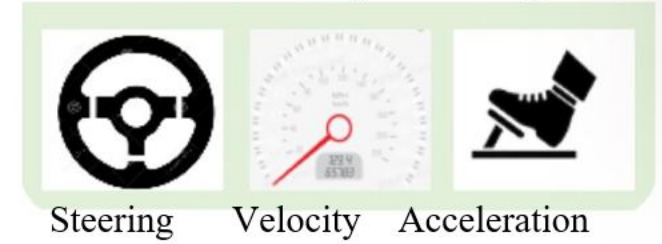
$x, y$	Coordinate of ego vehicle
$\Delta x^s, \Delta y^s$	Relative coordinate of ego and surrounding vehicle
$v, a_e$	Velocity and acceleration of ego vehicle
$\theta_e$	ego vehicle heading angle
$v_i^s$	surrounding vehicle velocity
$\Delta v_i^s$	Relative velocity of ego and surrounding vehicle
$\Delta d$	Relative distance between ego and leading vehicle
$t_{ttc}$	Time to collision

- $A_{steer}, A_{throttle}, A_{brake}$  are the normalized values of ego vehicle steering wheel angle, throttle, and brake, respectively.
- Ghost vehicle = (x,y) position of ego vehicle at future timestep.

## Action Space:



Continuous Action space for ego vehicle



Discrete Action for ego vehicle

$$A_1 = \text{Lane change action} = 1$$

$$A_2 = \text{Keep in lane action} = 0$$

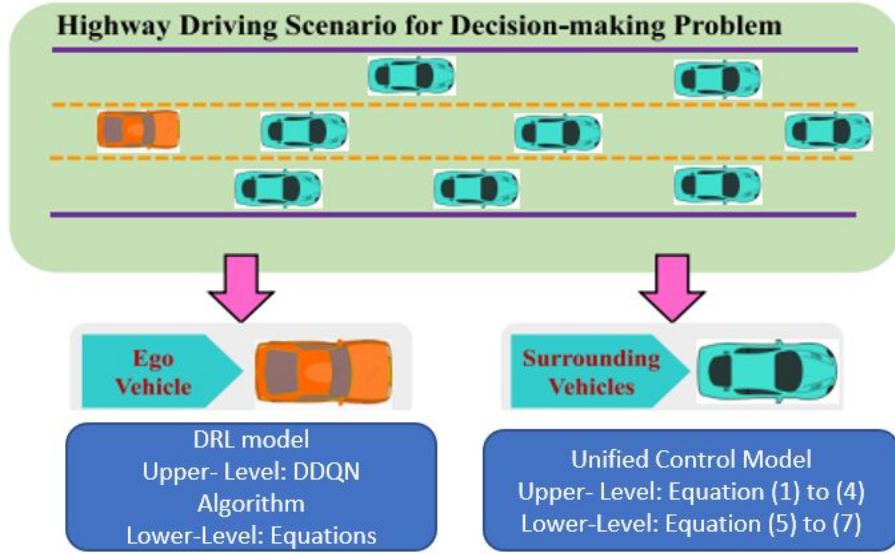
**Reward:** Rewards can be set based on vehicle progress matrices on the road without collisions. i.e lane change, keep in lane(L=1), collisions

$$\begin{cases} r_{collision} = -1 \\ r_{no\ collision} = -0.1 * (v_e^t - v_e^{max}) \\ r_{lane\ change} = -0.4 * (L - 1)^2 \end{cases}$$

$$r_{total} = r_{collision} + r_{no\ collision} + r_{lane\ change}$$



# Introduction: Highway Driving Scenario



$v, a$	current vehicle speed and acceleration
$a_{max}$	Maximum acceleration : $6 \text{ m/s}^2$
$d$	Distance to the front car
$\delta$	acceleration argument : 4
$d_{tar}, a_{tar}$	Target distance and Target acceleration
$d_0$	Predefined minimum relative distance : 10m
$T$	Expected time interval for safety goal : 1.5 s
$\Delta v$	Relative speed between two vehicles
$b$	Deceleration rate : $-5 \text{ m/s}^2$
$b_{safe}$	Maximum braking impose to follower in the lane change : $2 \text{ m/s}^2$
$z$	Politeness factor : 0.001

## Vehicle Behavior controller

Longitudinal behavior

$$a = a_{max} \left[ 1 - \left( \frac{v}{v_{tar}} \right)^\delta - \left( \frac{d_{tar}}{\Delta d} \right)^2 \right] \quad (1)$$

Since  $d_{tar}$  is affected by front vehicle:

$$d_{tar} = d_0 + Tv + \frac{v\Delta v}{2(\sqrt{a_{max}b})} \quad (2)$$

The safety criterion requires the follower in the desired lane (after changing) to limit its acceleration to avoid a collision

$$a_j^{new} \geq -b_{safe} \quad (3)$$

The incentive condition is imposed on the ego vehicle and its followers by an acceleration threshold  $a_{th}$

$$a_e^{new} - a_e^{old} + z[(a_i^{new} - a_j^{old}) + (a_j^{new} - a_j^{old})] > a_{th} \quad (4)$$

## Vehicle Motion Controller

• Acceleration controller  $\Rightarrow a = K_p(v_{tar} - v) \quad (5)$

• lateral speed controller  $\Rightarrow v_{lat} = -K_{p,lat}\Delta_{lat} \quad (6)$

• Heading control (yaw rate)  $\Rightarrow \dot{\phi} = K_{p,\phi}(\phi_{tar} - \phi) \quad (7)$

Where  $\Delta_{lat}$  is lateral position,  $\phi_{tar}$  is target heading angle,  $v_{tar}$  is target velocity

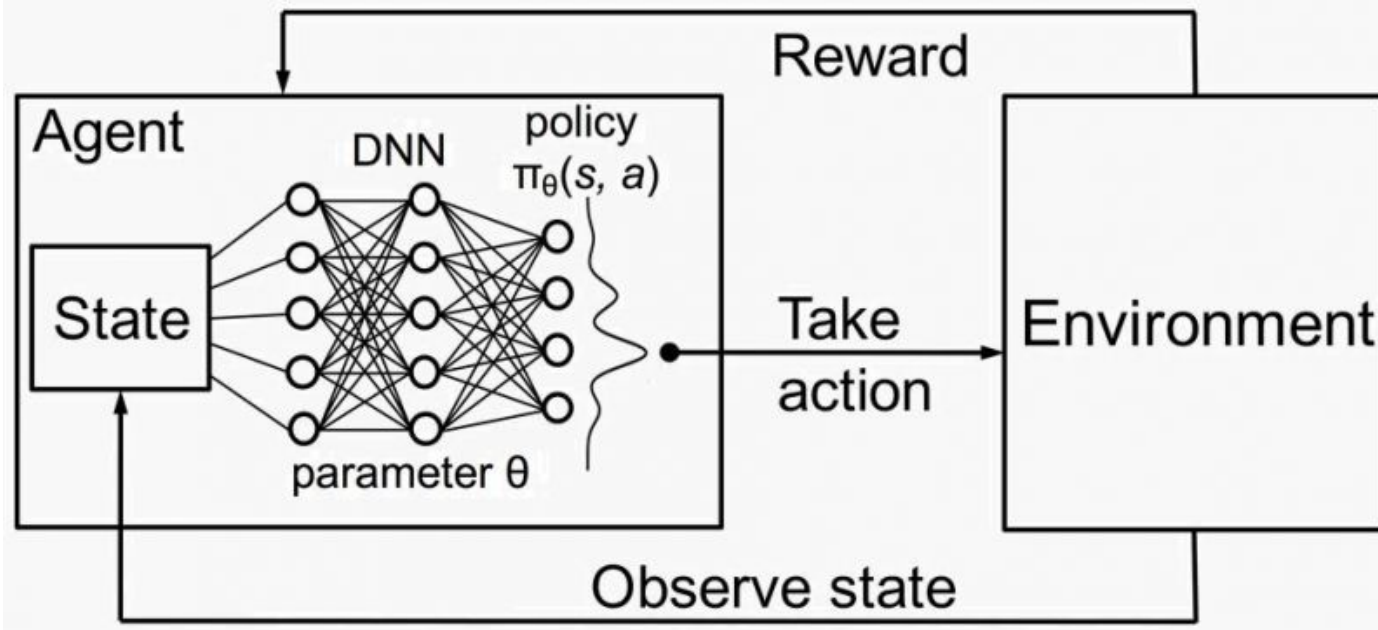
$K_p, K_{p,lat}, K_{p,\phi}$  are the proportional gains

\* the follower  $i$  (of the ego vehicle) at current lane, and the follower  $j$  at the target lane of lane change

# Deep Q-Network (DQN)

The common DRL methods are unable to address the highway overtaking problems because of the continuous action space and large state space.

**Deep Q-Network (DQN)** utilizes neural networks to approximate the Q-table. The neural takes state and action space as its inputs and outputs state-value function.



DQN Architecture

## Deep Q-Network

### Recursive Step:

$$Q(s, a; \theta) \leftarrow Q(s, a) + \alpha [r + \underbrace{\gamma \max_{a'} Q(s', a'; \theta')}_{\text{Temporal Difference}} - Q(s, a; \theta)]$$

### Loss Function:

$$L(\theta) = E[\sum_{t=1}^N (y_t - Q(s, a; \theta))^2]$$
$$y_t = r_t + \gamma \max_{a'} Q(s', a'; \theta')$$

### Gradient Descent Step:

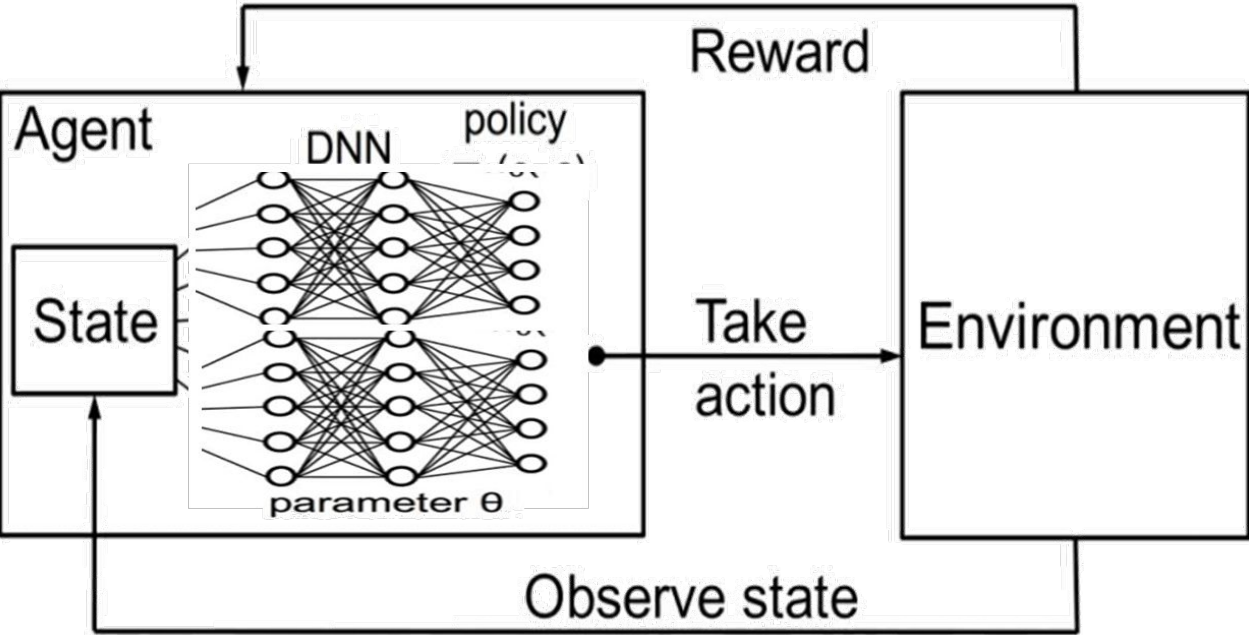
$$\nabla_{\theta} L(\theta) = E[(y_i - Q(s, a; \theta)) \nabla_{\theta} Q(s, a; \theta)]$$

Here  $s'$  and  $a'$  are the state and action at the next time step.

$\theta$  and  $\theta'$  are the parameters of neural network.

# Double-Deep Q-Network

**Double Deep Q-Network (D-DQN)** is similar to DQN main difference is D-DQN uses **two identical neural network models**, one learns during the experience replay similar to DQN, and the other copy the last episode of the first model.



D-DQN Architecture

It uses secondary model that is the copy of the main model from the last episode. Since the difference between value of the second model are lower than the main model, we use this second model to attain the Q value.

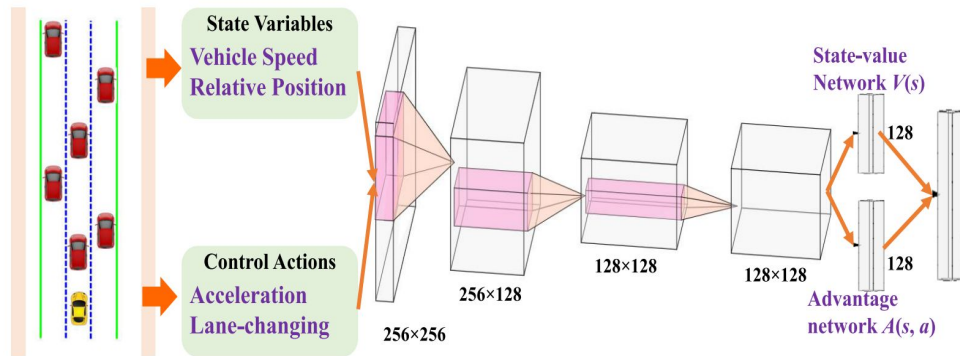
**Bellman Equation for Double DQN**

$$Q(s, a; \theta) = r + \gamma Q(s', \operatorname{argmax}_{a'} Q(s', a'; \theta'); \theta')$$

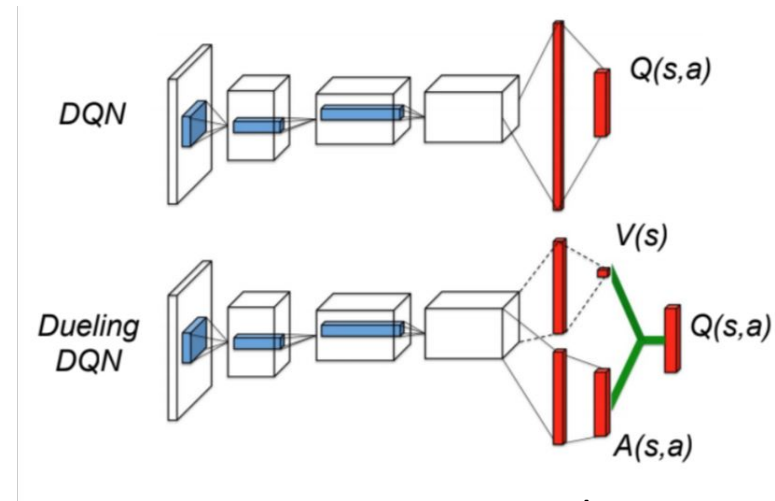
# Dueling-Deep Q-Network (DDQN)

DQN Reinforcement learning method tends to overestimate the rewards because it only uses the maximum value. In this project, it means that the DQN algorithm may not produce negative rewards for cases that would cause accident.

**Dueling Deep Q-Network** eliminates this problem by splitting the neural network into two branches, one estimates the states and the other estimates the advantage function and combine the results later.



DDQN Architecture implemented



DQN vs DDQN Architecture

## Recursive Step:

$$Q^{\pi}(s, a; \theta) = V^{\pi}(s; \theta_1) + A^{\pi}(s, a; \theta_2)$$

$$Q^{\pi}(s, a; \theta) = V^{\pi}(s; \theta_1) + (A^{\pi}(s, a; \theta_2) - \max_{a'} A^{\pi}(s, a'; \theta_2))$$

## Advantage Function:

It determines how much a certain action is good or bad given a particular state.

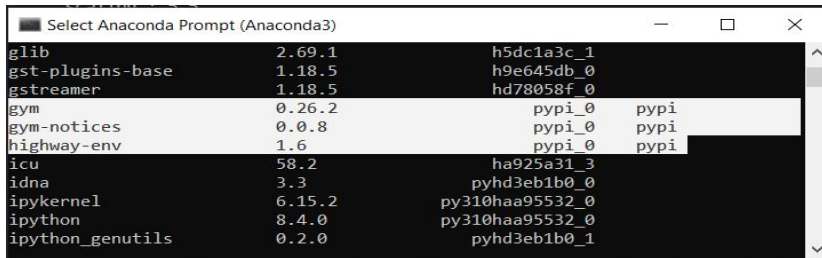
$$A(s, a) = Q(s, a) - V(s)$$



# Implementation and Simulation

```
(rl-env-eee598) C:\Users\sjain198\rl-eee598>conda info --env
# conda environments:
#
base                  C:\Users\sjain198\Anaconda3
LaneChange            C:\Users\sjain198\Anaconda3\envs\LaneChange
eee598-rl-env         C:\Users\sjain198\Anaconda3\envs\eee598-rl-env
lanechange            C:\Users\sjain198\Anaconda3\envs\lanechange
marl_cav              C:\Users\sjain198\Anaconda3\envs\marl_cav
radar                 C:\Users\sjain198\Anaconda3\envs\radar
rl-env-eee598         * C:\Users\sjain198\Anaconda3\envs\rl-env-eee598
```

Successfully created the conda simulation environment



Package Name	Version	Channel
glib	2.69.1	h5dc1a3c_1
gst-plugins-base	1.18.5	h9e645db_0
gststreamer	1.18.5	hd78058f_0
gym	0.26.2	pypi_0
gym-notices	0.0.8	pypi_0
highway-env	1.6	pypi_0
icu	58.2	ha925a31_3
idna	3.3	pyhd3eb1b0_0
ipykernel	6.15.2	py310haa95532_0
ipython	8.4.0	py310haa95532_0
ipython_genutils	0.2.0	pyhd3eb1b0_1

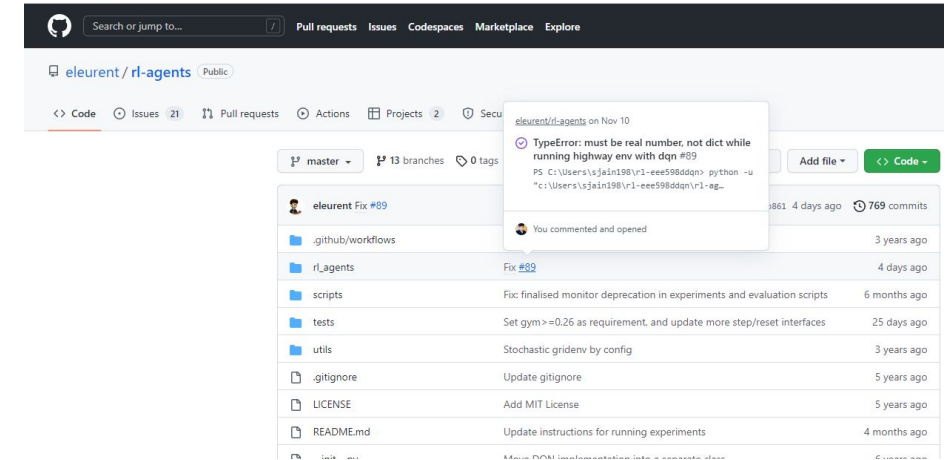
Successfully install the highway environment in rl-env-ee598



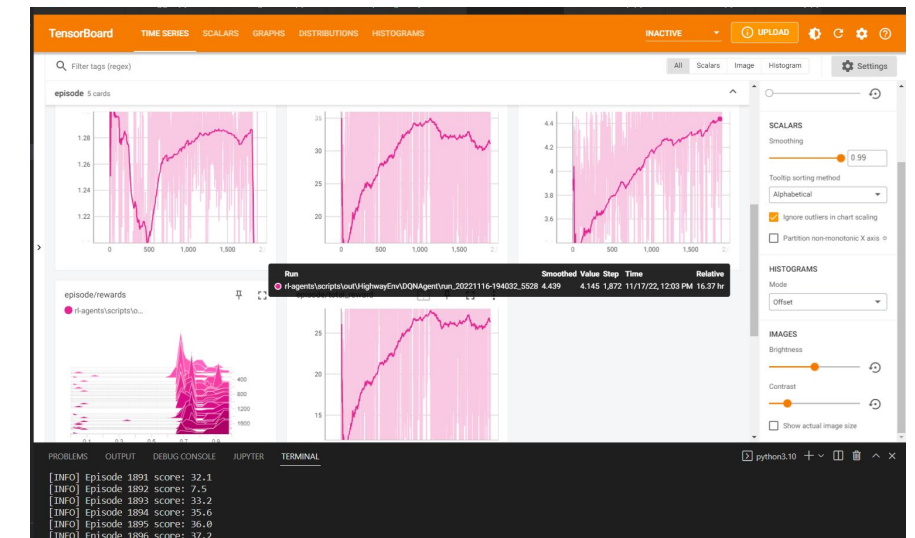
We deployed our algorithm in the highway-env for simulation

## Requirements

- System requires (Recommend)
  - GPU  $\geq$  Nvidia GTX 3050Ti
  - CPU  $\geq$  Intel i7 vPRO
  - RAM  $\geq$  4 GB
- Programming Language
  - Python  $\geq 3.5$
- Setup requires
  - pytest-runner
  - rl-agent
- Install requires
  - gym  $\geq 0.26$
  - numpy
  - pygame  $\geq 2.0.2$
  - matplotlib
  - pandas
  - Scipy
  - Numba
  - seaborn
  - six
  - docpt
  - torch  $\geq 1.2.0$
  - tensorboardX



We contributed to fix the main code.



It took 18h for training for 30s drive



# Results : Driving Conditions

Figure 1, figure 2 and figure 3 are some typical testing driving condition: the ego vehicle make a dangerous lane changing and a collision happens.

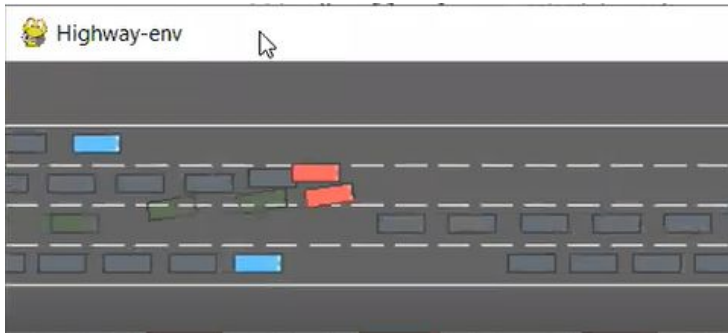


Figure 1

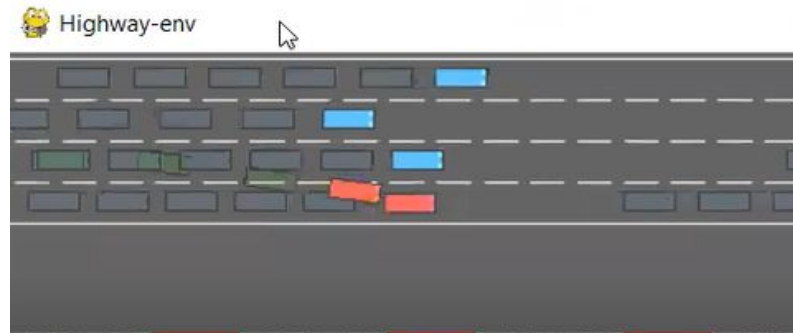


Figure 2

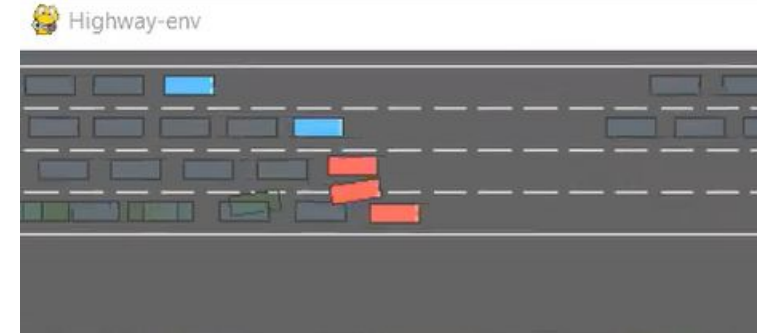


Figure 3



Figure 4

Figure 4 is another representative testing driving condition: the ego vehicle make wrong lane change.



Figure 6

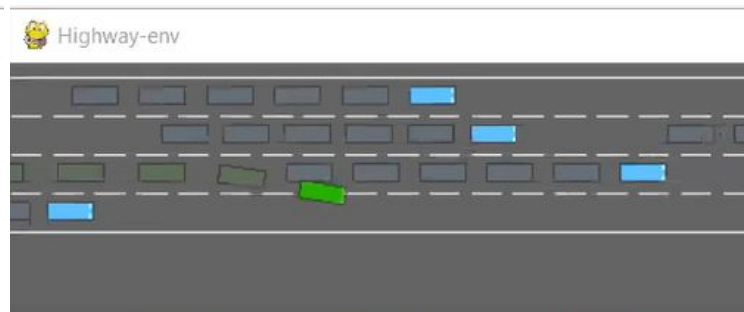


Figure 7

Figure 6 and figure 7 are the driving conditions after learning when vehicle start make good decisions .

# Results

## Results:

- A higher traveling distance means the ego vehicle could drive longer during a time limit without collision. Dueling DQN has greatest length travelled for any given number of episodes.
- It is also obvious that the training stability and learning speed of Dueling DQN are better than the rest
- The reward in Dueling DQN is greater than the other DQN approaches, and it keeps this momentum all the time.
- Higher reward indicates driving on the preferred lane with a more efficient maneuver.

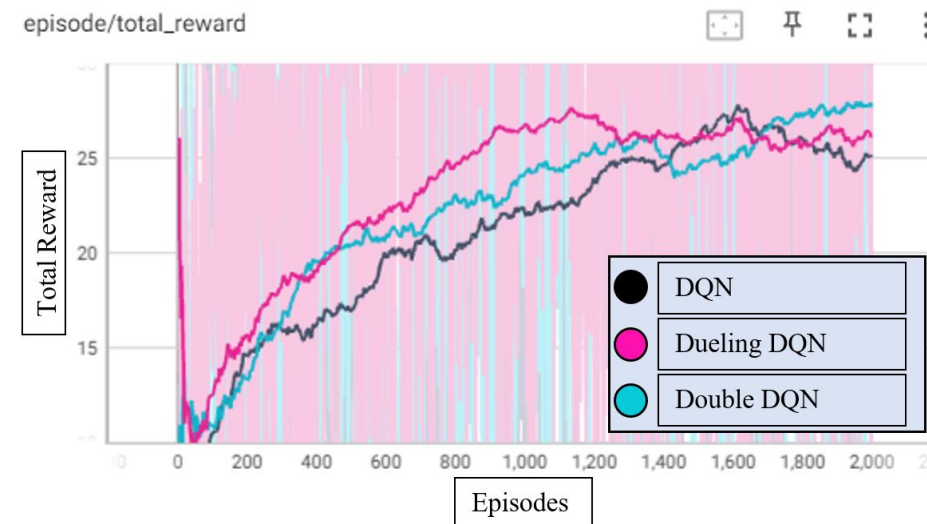


Figure: Total Reward vs Episode

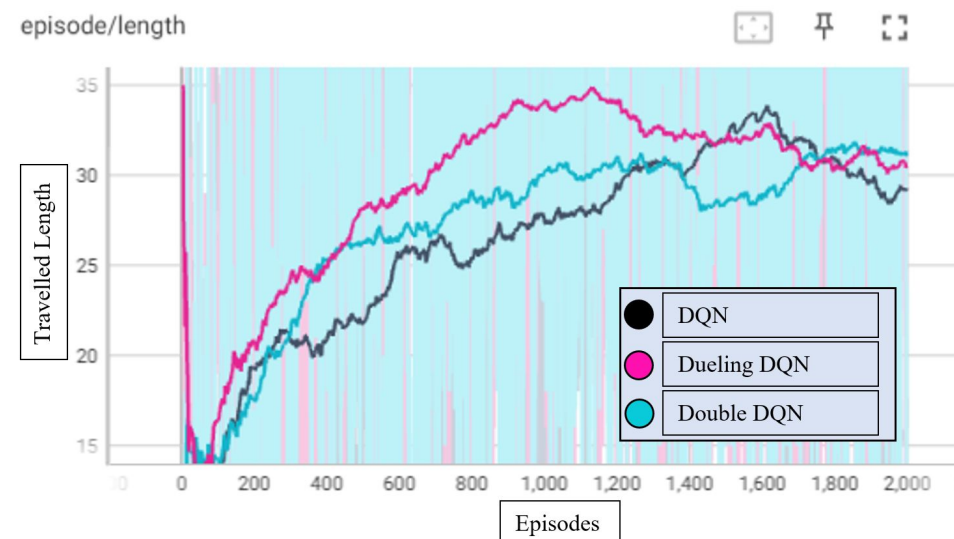


Figure: Total Length vs Episode

# Results



## Simulation DQN

```
1 {
2   "environments": ["configs/highway/env.json"],
3   "agents": [
4     "configs/highway/env/agents/DQNAgent/dqn.json"
5   ]
6 }
7
```

```
PS C:\Users\sjain\Documents\Decision-Making-Strategy-on-Highway-for-Autonomous-Vehicles> python experiments.py benchmark Multiple_agents.json --test --processes=4 --episodes=1 --recover-from C:\Users\sjain\Documents\Decision-Making-Strategy-on-Highway-for-Autonomous-Vehicles\scripts\slout\highwayenv\QDQAgent\run_20221116-194032_16060\checkpoint_best.tar
```

Total Reward: 61.7  
Video: [Click here](#)

## Simulation Double DQN

```
1 {
2   "environments": ["configs/highway/env.json"],
3   "agents": [
4     "configs/highway/env/agents/DDQNAgent/ddqn.json"
5   ]
6 }
7
```

```
PS C:\Users\sjain\Documents\Decision-Making-Strategy-on-Highway-for-Autonomous-Vehicles> python experiments.py benchmark Multiple_agents.json --test --processes=4 --episodes=1 --recover-from C:\Users\sjain\Documents\Decision-Making-Strategy-on-Highway-for-Autonomous-Vehicles\scripts\slout\highwayenv\DDQAgent\run_20221116-194032_16060\checkpoint_best.tar
```

Total Reward: 68.6  
Video: [Click here](#)

## Simulation Dueling DQN

```
1 {
2   "environments": ["configs/highway/env.json"],
3   "agents": [
4     "configs/highway/env/agents/DDQNAgent/ddqn.json"
5   ]
6 }
7
```

```
PS C:\Users\sjain\Documents\Decision-Making-Strategy-on-Highway-for-Autonomous-Vehicles> python experiments.py benchmark Multiple_agents.json --test --processes=4 --episodes=1 --recover-from C:\Users\sjain\Documents\Decision-Making-Strategy-on-Highway-for-Autonomous-Vehicles\scripts\slout\highwayenv\DDQAgent\run_20221116-194032_16060\checkpoint_best.tar
```

Total Reward: 73.5  
Video: [Click here](#)

# Conclusion and lessons learned



- Finally, from this project we were able to develop a much better understanding of the three widely Reinforcement Learning networks, DQN, Dueling DQN and Double DQN.
- We were also able to experience, how a machine learns to do a complicated task such as lane changing and see in real time how it makes improvements in it's approach.
- While, trying to code we encountered problems like mismatch in the package versions and tensor multiplication error. We learned how to handle a number of tensors and other matrices simultaneously and work operations on them.
- We also found bugs in the open-source package that was hindering our progress. So we corrected the bugs and contacted the publisher of the open source package, who then made the necessary changes.
- We also learned how to evaluate the effectiveness of different algorithms using benchmark factors such as Total Reward and the total length traveled for given number of episodes trained.



- **Evidences key literature:**
  - **Paper** : J. Liao, T. Liu, X. Tang, X. Mu, B. Huang and D. Cao, "Decision-Making Strategy on Highway for Autonomous Vehicles Using Deep Reinforcement Learning," in IEEE Access, vol. 8, pp. 177804-177814, 2020, doi: 10.1109/ACCESS.2020.3022755.
  - **Book Reference** : R. S. Sutton and A. G. Barto. Reinforcement Learning: an Introduction, 2018
- **Evidence of simulation environment and rl-agent libraries :**
  - Highway Environment ([reference link](#))
  - OpenAI gym ([reference link](#))
  - Stable\_baseline3([reference link](#))
  - eleurent rl-agent ([reference link](#))
- **Evidence of implementation code :**
  - We implemented DDQN, DQN, Dueling DQN([reference link](#))

# Thank You!

A solid yellow horizontal bar located below the "Thank You!" text.