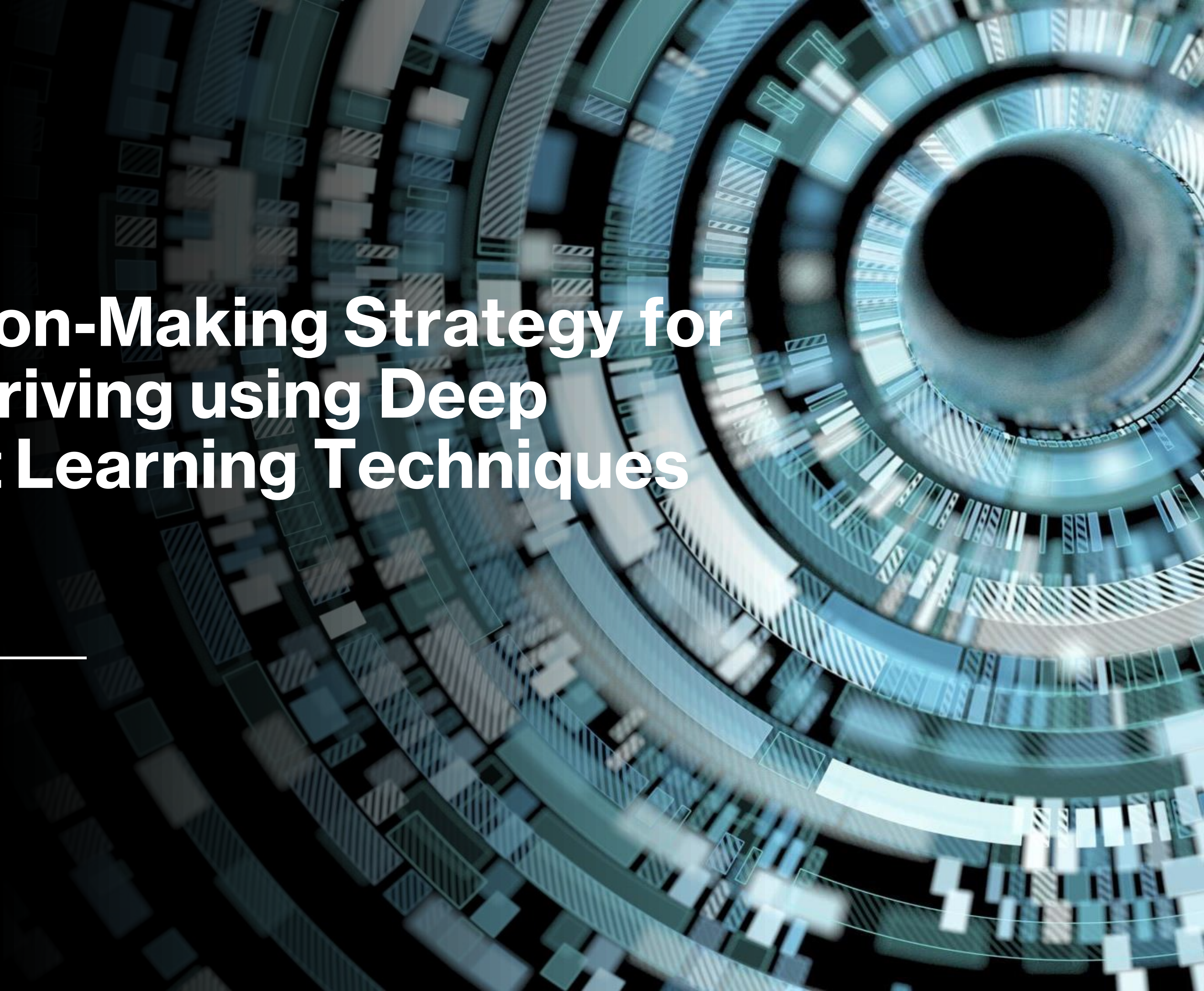
Tactical Decision-Making Strategy for Autonomous Driving using Deep Reinforcement Learning Techniques

Team Members

Sivaranjani Kumar

Vigneshkumar Thangarajan



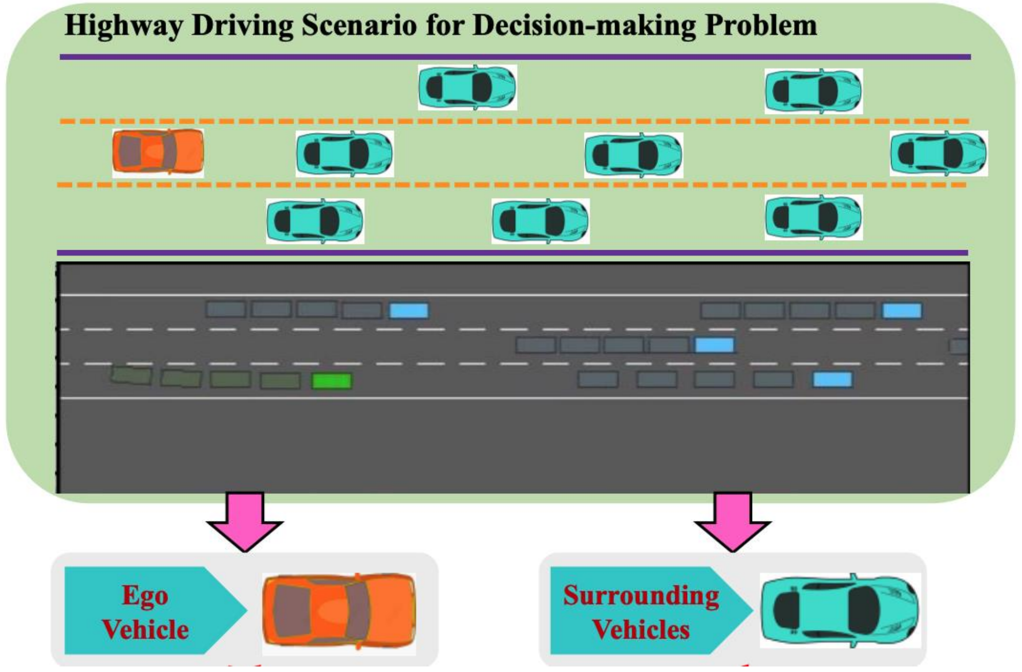
Introduction

The cost of human driving data collection at large scale can be expensive. The promising approach is to train a policy using reinforcement learning. The promise of leveraging data to automatically learn a complex driving policy to imitate the human driving decisions is phenomenal.

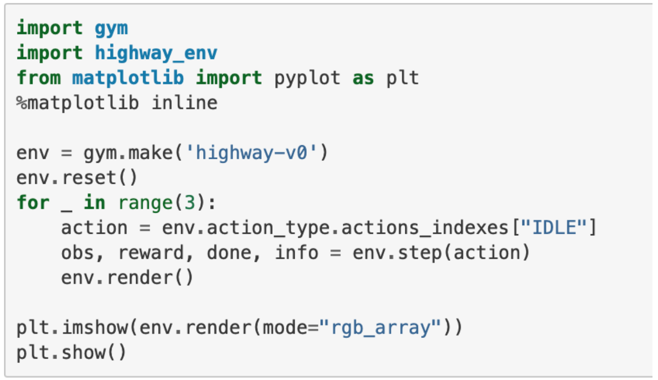
Goal : Train an autonomous driving agent to drive safely and efficiently in a simulated highway environment.

Deep reinforcement learning methods like DQN and DDQN with Dueling Network architectures are implemented to improve the driving efficiently.

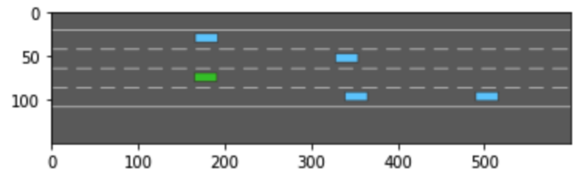
Reinforcement Learning Terminologies

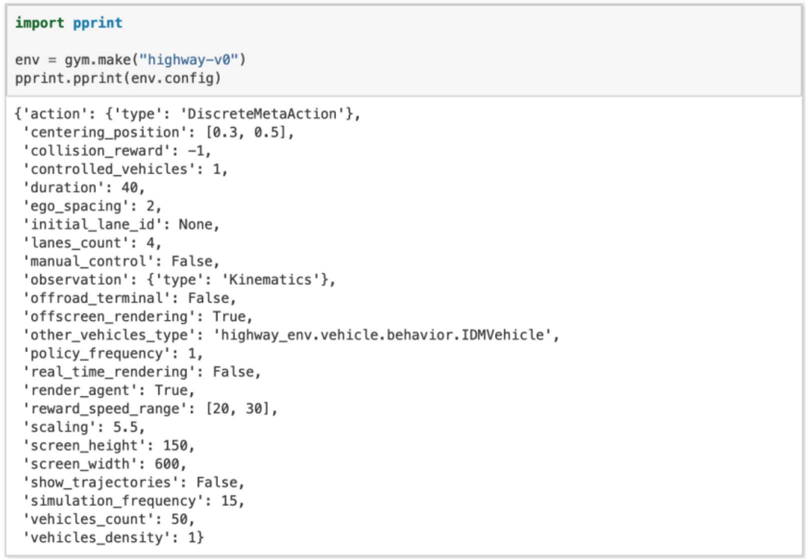


- Environment – Highway-env
- Agent – Ego Vehicle
- Action - Lane Left, Lane Right. Idle, Faster, Slower.
- Observation - Environment emits the observation Ot+1 and received by agent Ot
- Reward – Positive value for avoiding collision (1) and negative value for collision (-1).



```python
import gym
import highway_env
from matplotlib import pyplot as plt
%matplotlib inline

env = gym.make('highway-v0')
env.reset()
for _ in range(3):
    action = env.action_type.actions_indexes["IDLE"]
    obs, reward, done, info = env.step(action)
    env.render()

plt.imshow(env.render(mode="rgb_array"))
plt.show()
```





```python
import pprint

env = gym.make("highway-v0")
pprint.pprint(env.config)
```

```
{'action': {'type': 'DiscreteMetaAction'},
 'centering_position': [0.3, 0.5],
 'collision_reward': -1,
 'controlled_vehicles': 1,
 'duration': 40,
 'ego_spacing': 2,
 'initial_lane_id': None,
 'lanes_count': 4,
 'manual_control': False,
 'observation': {'type': 'Kinematics'},
 'offroad_terminal': False,
 'offscreen_rendering': True,
 'other_vehicles_type': 'highway_env.vehicle.behavior.IDMVehicle',
 'policy_frequency': 1,
 'real_time_rendering': False,
 'render_agent': True,
 'reward_speed_range': [20, 30],
 'scaling': 5.5,
 'screen_height': 150,
 'screen_width': 600,
 'show_trajectories': False,
 'simulation_frequency': 15,
 'vehicles_count': 50,
 'vehicles_density': 1}
```

Highway Environment

- The ego-vehicle is driving on a multilane highway populated with other vehicles. The agent's main objective is to drive efficiently by following safe maneuvers while avoiding collisions with neighboring vehicles.
- The vehicle is driving on a straight highway with several lanes, and is rewarded for reaching a high speed, staying on the rightmost lanes and avoiding collisions.

RIGHT_LANE_REWARD*: float= 0.1*

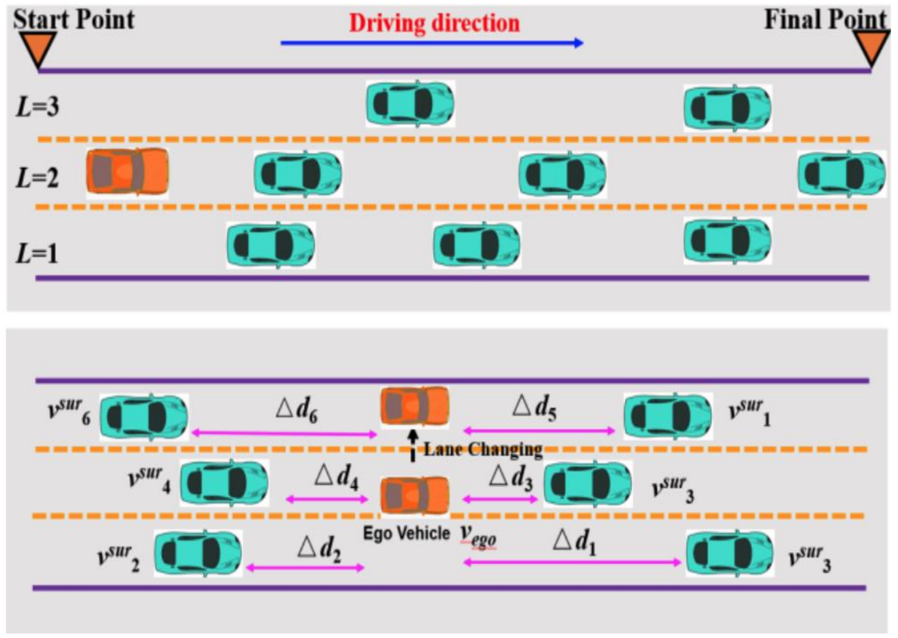
HIGH_SPEED_REWARD*: float= 0.4*

LANE_CHANGE_REWARD*: float= 0*

COLLISION_REWARD*: -1*

- The observations, actions, rewards and dynamics of an environment are parametrized by a configuration, defined as a **config** dictionary.
- For highway environments, several types of observations can be used. They are defined in the observation module.
- Kinematics, grayscale, occupany grid and time to collision.

Driving Behavior



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

- The action is executed by the ego vehicle and all other surrounding vehicles exhibit the default behavior for several simulation timesteps and they deigned randomly to create uncertainties in the driving environment.
- Hierarchical Motion Controller: To manage the lateral and longitudinal movements of the ego and surrounding vehicles.
  - The upper-level contains two models, which are Intelligent driver model (IDM) and Minimize overall braking induced by lane changes (MOBIL).
  - The lower-level focuses on regulating vehicle velocity and acceleration.
  - The original speed of the ego vehicle is chosen from [23, 25] m/s.
  - Maximum speed is 40 m/s.
  - The length and width of all vehicles are 5m and 2m.
  - The initial velocity of the surrounding vehicles is randomly chosen from [20, 23] m/s, and their behaviors are manipulated by IDM and MOBIL.
  - V and a – current speed and acceleration.
  - amax - Desired speed.
  - Vtar and dtar – target velocity and distance

Motivation to DQN

1. Traditional RL methods has a shotcomings and they limits to discrete action spaces and cannot solve continuous control problems.

2. Deep Reinforcement Learning (DRL) are the powerful tool to deal with long sequential problems and it is independent of the historical driving data. But DRL was unable to address the highway overtaking problems because of the continuous action space and large state space.

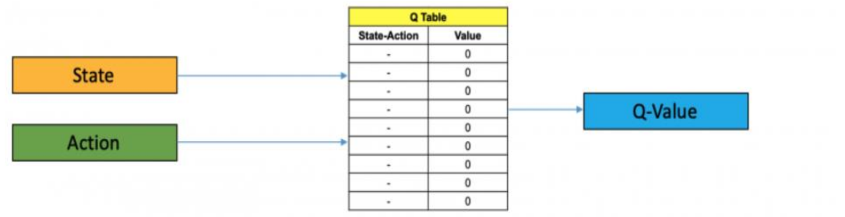
3. Deep neural network maps the state and action into a value.

4. Motivation for Deep Q Network to solve this problem is – huge number of states causing numerous combinations of possible state-action combinations.

5. Highway environment is ever changing and requires lot of time to explore all the state.

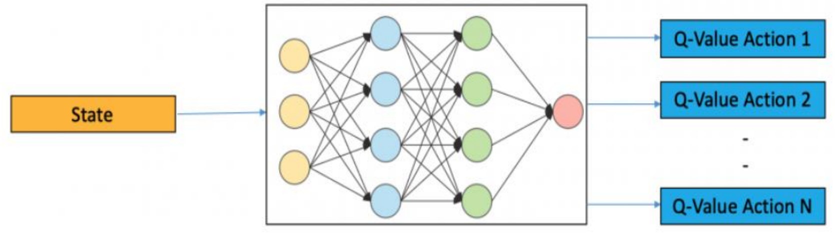
Deep Q Learning

- Q-learning: Reward projected into future after an action is taken place and its explorations are decoupled from the actions.
- Deep Q-learning: Deep Neural networks are the function approximators, where state is given as input and separate outputs for each possible action.
- To overcome the non - I.I.D attribute of input data, DQN uses experience replay buffer and target network to predict Q values.
- **Experience Replay –** It stores experiences including state transitions, rewards and actions, which are necessary data to perform Q learning, and takes random mini-batches to update neural networks.

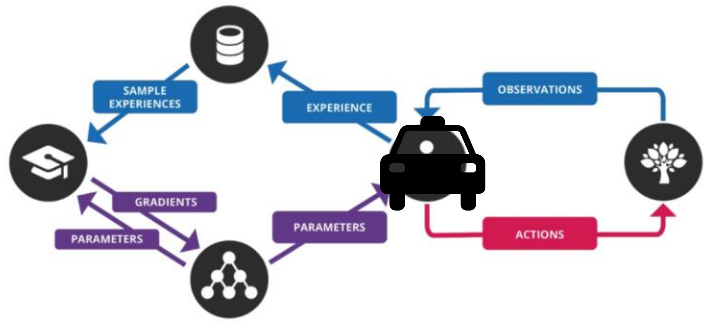


State-Action	Value
-	0
-	0
-	0
-	0
-	0
-	0
-	0
-	0
-	0

Q Learning



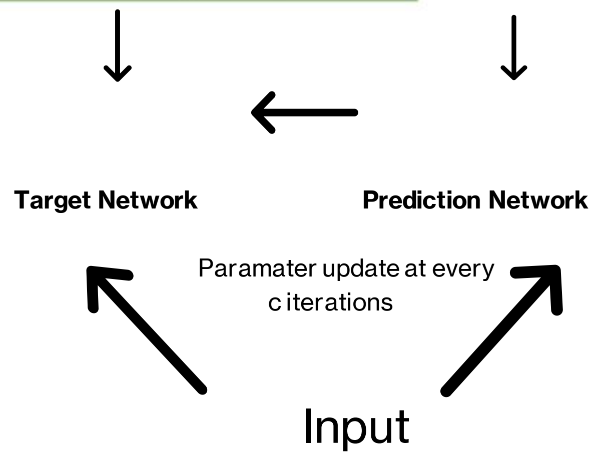
Deep Q Learning



Double DQN

- Since the target Q value continuously changes with each iteration, in DDQN we have two identical networks – one to learn the Q value and the other to learn the target value.
- **Target Network and Prediction network -** Using separate network to estimate the target and has the same architecture as function approximator. After c iterations, the parameters from the prediction network are passed to target network. This leads to stable training because it keeps the target functions fixed.

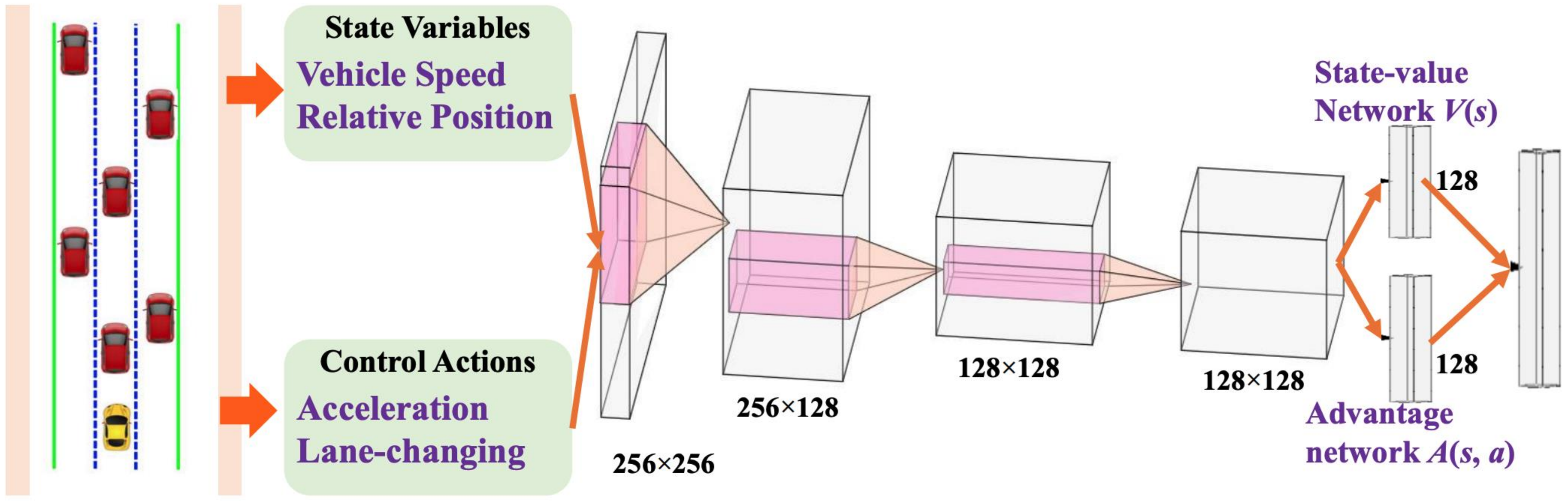
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[\boxed{R_{t+1} + \gamma \max_a Q(S_{t+1}, a)} - Q(S_t, A_t)\right]$$



Dueling DQN Intuition

- Dueling DQN has two separate network one to estimate the state-value and the other one to estimate the advantages for each action in that state.
- This solves one of the specific problem to this domain. In environments like Atari single agent games and highway environment like this, at most of the states the agent action is not relevant.
- For example, When there are no chance of collision in any lane, the agent action is of least important. Dueling architecture addresses the problem of Q value overshooting and prevents model to learn from high Q values.

The Dueling Network



Q value function for DDQN

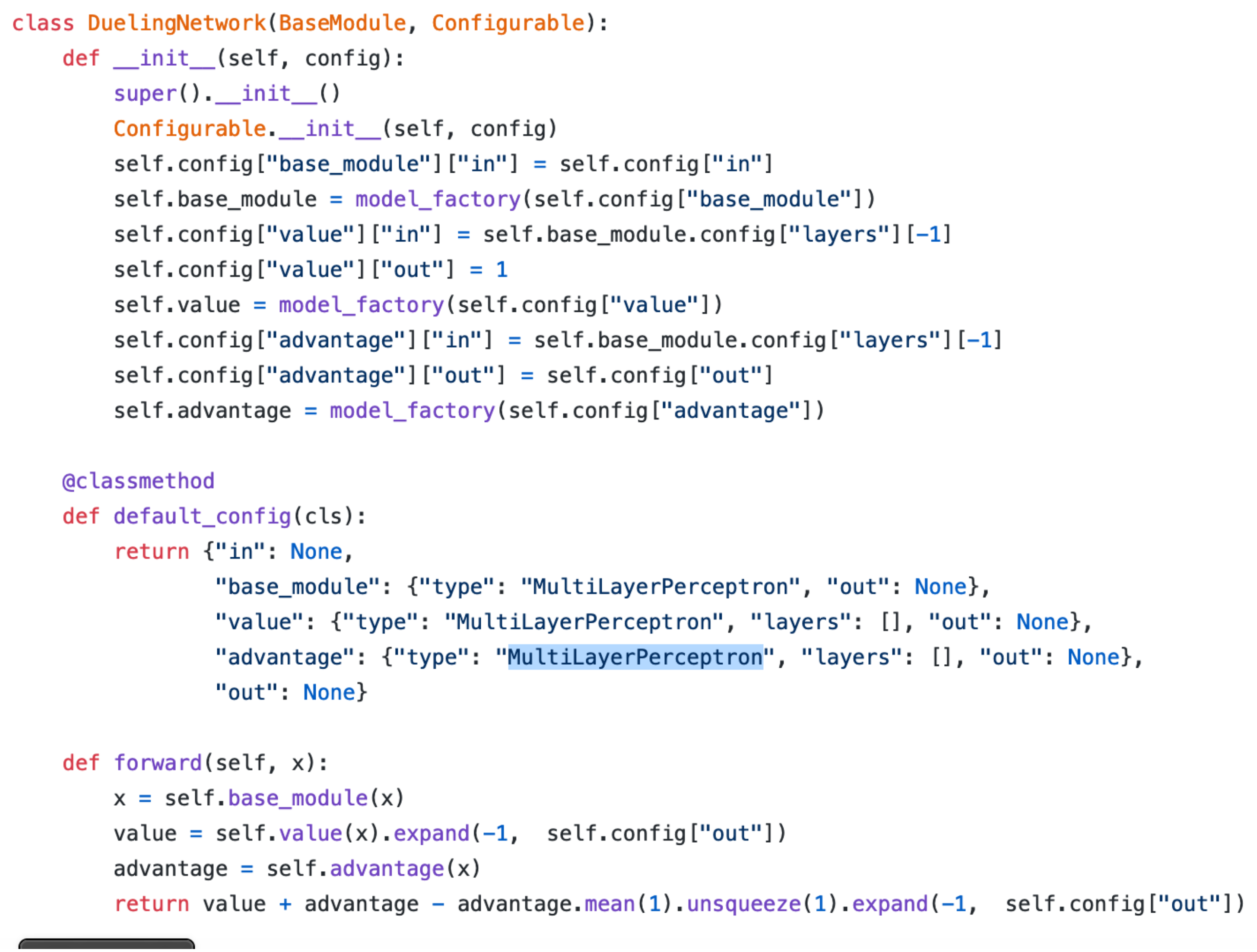
- Theta – parameters of conv layers
- Alpha, Beta – parameters of fully connected layers
- The subtract by mean part also solves the problem of un-identifiability. Thus, we can now recover function V and A separately given Q.
- This equation is implemented as part of the network and not as a separate algorithm.
- Thus, we can accommodate this network with any RL model like DDQN, SARSA etc.

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s,a';\theta,\alpha) \right).$$

Dueling DDQN Hyperparameters

- "gamma": 0.8,
- "n_steps": 1,
- "batch_size": 32,
- "memory_capacity": 15000,
- "target_update": 50,
- "exploration"
  - "method": "EpsilonGreedy",
  - "tau": 6000,
  - "temperature": 1.0,
  - "final_temperature": 0.05
- "loss_function": "l2"

Dueling Network Implemention



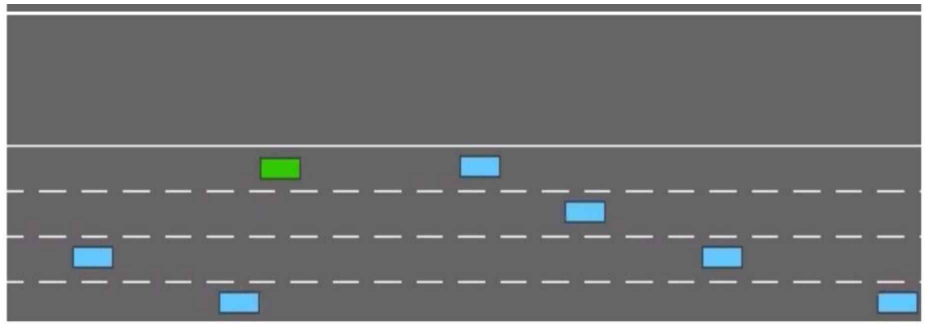
```python
class DuelingNetwork(BaseModule, Configurable):
    def __init__(self, config):
        super().__init__()
        Configurable.__init__(self, config)
        self.config["base_module"]["in"] = self.config["in"]
        self.base_module = model_factory(self.config["base_module"])
        self.config["value"]["in"] = self.base_module.config["layers"][-1]
        self.config["value"]["out"] = 1
        self.value = model_factory(self.config["value"])
        self.config["advantage"]["in"] = self.base_module.config["layers"][-1]
        self.config["advantage"]["out"] = self.config["out"]
        self.advantage = model_factory(self.config["advantage"])

    @classmethod
    def default_config(cls):
        return {"in": None,
                "base_module": {"type": "MultiLayerPerceptron", "out": None},
                "value": {"type": "MultiLayerPerceptron", "layers": [], "out": None},
                "advantage": {"type": "MultiLayerPerceptron", "layers": [], "out": None},
                "out": None}

    def forward(self, x):
        x = self.base_module(x)
        value = self.value(x).expand(-1,  self.config["out"])
        advantage = self.advantage(x)
        return value + advantage - advantage.mean(1).unsqueeze(1).expand(-1,  self.config["out"])
```

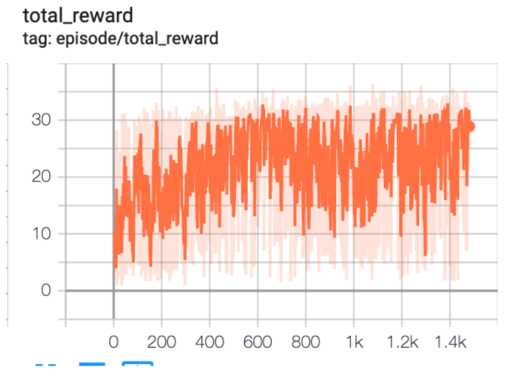
Model Training

- Training is done in Google colab Pro account for 2000 episodes each for plain DQN and Dueling DQN model.
- It takes ~8 hours to train one model.
- With plain DQN, we can see in one of the trials the car crashes even after training for 2000 episodes.
- The high variance of total rewards is explicitly visible in case of DQN whereas we can see lower variances in case of Dueling DQN, and no crash observed.

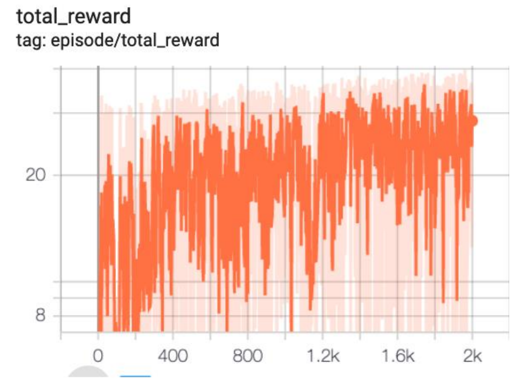
Results & Conclusion

- After training for 2000 episodes, the simulation results shows that DQN with Dueling network architecture performs better than DQN.
- In future, we can try Noisy DQN model and DQN with Prioritized Experience Replay to see if we can improve the current model performance


Snapshot of trained agent in Simulation



Deep Q-Network



Dueling Deep Q-Network

References

- [1] Liao, Jiangdong., Liu, Teng., Tang, Xiaolin., Mu, Xingyu., Huang, Bing., & Cao, Dongpu, "Decision-making strategy on Highway for Autonomous Vehicles using Deep Reinforcement Learning"
- [2] Li, Xin., Xu, Xin., Zuo, Lei, "Reinforcement Learning based overtaking decision-making for highway autonomous driving" in 2015 Sixth International Conference on Intelligent Control and Information Processing (ICICIP)
- [3] L. Edouard, "An environment for autonomous driving decisionmaking," https://github.com/ eleurent/highway-env, GitHub, 2018.
- [4] M. Zhou, X. Qu, and S. Jin, "On the impact of cooperative autonomous vehicles in improving freeway merging: a modified intelligent driver model-based approach," IEEE Trans. Intell. Transport. Syst., vol. 18, no. 6, pp. 1422-1428, June 2017

Questions ?

Thank you