Weekly Report

ME17B179

The video driver in the DCL_PC_2 crashes when I open CARLA remotely through anydesk, therefore I wasn't able to stabilize the Deep-Q learning algorithm for the task where the car must successfully manoeuvre the straight road by evading 2 obstacles on the straight road. I hope I can fix the video driver issue in person.

Prof. Ramakrishna suggested to look at the link which is another environment that very much comes under the autonomous car navigation domain.

The name of this open-source simulator/environment is highway-env - https://github.com/eleurent/highway-env

This github has a collection of environments for autonomous driving and tactical decision-making tasks. These environments allow customizations to certain extent. My aim was to solve the first environment which offers a task where we need to control the vehicle on a straight road to move forward by evading moving obstacle vehicles by changing lanes.

highway-v0 Environment Details

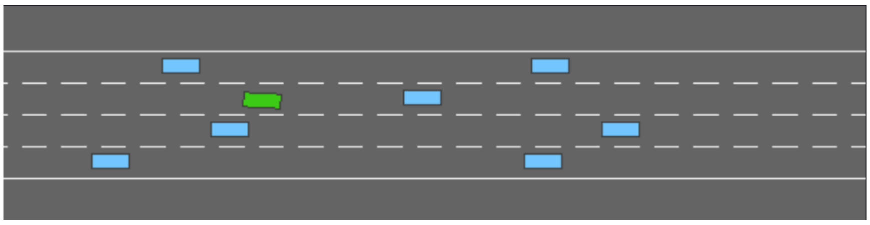

Fig 1. Highway environment simulator

Aim of the environment is to avoid moving obstacles on a straight road with 4 lanes. The agent and the obstacles vehicle all move in the same direction and can slow down or speed up or change lanes. The agent vehicle is represented as green box and other obstacles are represented as blue boxes.

Dynamics of the Agent Vehicle

The vehicles follow simple kinematic bicycle model

- $\dot{x} = v (\cos (\varphi + \beta))$
- $\dot{y} = v (\sin (\varphi + \beta))$
- $\dot{v} = a$
- $\dot{\varphi} = (v/l) \sin(\beta)$
- $\beta = \tan^{-1} (\tan(\delta)/2)$

where x, y is the vehicle position, v is its forward speed, φ is heading angle, a is the acceleration command, β is the slip angle at the centre of gravity, δ is the front wheel angle.

Observation of the Agent Vehicle

There are 4 types of observations provided by the environment

- Kinematics
- Greyscale Image
- Occupancy Grid
- Time to collision

I have majorly experimented with the kinematics and occupancy grid observations

The Kinematic Observation is a V×F array that describes a list of V nearby vehicles by a set of features of size F, listed in the "features" configuration field. The features are:

- x coordinate
- y coordinate
- v_x
- v_y

The OccupancyGrid Observation is a F×W×H array, that represents a grid of shape W×H discretising the space (x,y) around the ego-vehicle in uniform rectangle cells. Each cell is described by F features, listed in the "features" configuration field. The presence feature is the most useful one, it presents us with a matrix that discretizes the space relative to the agent vehicle and has either value 1 for obstacle or value 0 for empty space.

Actions of the Agent Vehicle

- 0 – Move to the adjacent left lane
- 1 – Remain idle
- 2 – Move to the adjacent right lane
- 3 – Accelerate
- 4 – Decelerate

Some of these actions might not be always available like lane changes at the edges of the roads, or accelerating/decelerating beyond the maximum/minimum velocity is automatically ignored by the environment. These cases are handled by the environment by continuing the idle action.

The discrete actions mentioned above are high level actions, the lane changing manoeuvres are already prebuilt into the system of low level controls.

Note: We can also customize the environment to use low-level continuous actions to control throttle accelerations and steering wheel angles.

Rewards

- $v_f = v \times \cos(\delta)$
- $a = 0.4 \times (v_f - v_{min})/(v_{max} - v_{min})$
- $b = 0.1 \times (l/n_l)$
- $c = d \times -1$, where $d = 1$ if collision, else $d = 0$
- Reward = $(a+b+c+1)/1.5$

Where v_f, v_{min}, v_{max} are the forward, minimum and maximum speed of the agent vehicle respectively. l represents the lane number and n_l is total number of lanes on the road. Usually $v_{min} = 20$, $v_{max} = 30$, $n_l = 4$. The range of reward spans between 0 and 1.

The episode terminates when the agent crashes or it exceeds 40 timesteps of simulation.

Experimentation

Both DQN and PPO RL algorithms has been implemented and both these algorithms show solid learning curves. Testing method is also implemented which collects 5 episodes of video recordings of the simulations for the best model saved during training process.

PPO

PPO has been run for 300 seasons, where each season consists of 10 episodes. At each season's end, the trajectories are collected and trained once. Two best variations of trained results are as follows:

Variation 1:

"Vehicle density" is a customizable parameter of the environment which is by default set to 1.0, as we increase the vehicle density the number of obstacles on the road around the agent increases. In this variation, we use the vehicle density to be 1.0 and use kinematics observation.

Result:

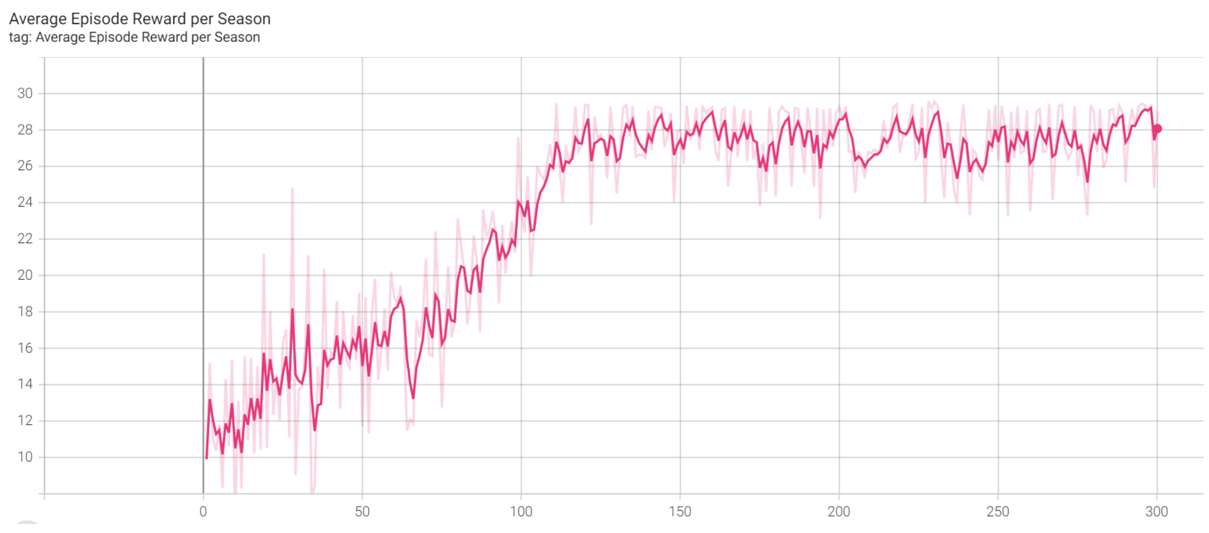

Fig 2. Average Episode Reward per Season vs nth Season

We can observe a learning curve which rises until around 125 episodes and saturates for further episodes.

Training data and Test Simulation: Drive

As the vehicle density is 1.0 which is low, the agent learns to slow down to the obstacles until the time limit exceeds rather than overtaking them. Less number of cars around the agent makes it hard to collect samples with crashes for the agent to learn better.

Variation 2:

Vehicle density is set to 1.5 in order to increase the complexity of the environment and force the agent to learn to overtake and change lanes rather than decelerating. Instead of randomly gathering the kinematic features of the nearest cars, we use occupancy grid observation with only "presence" information.

Result:

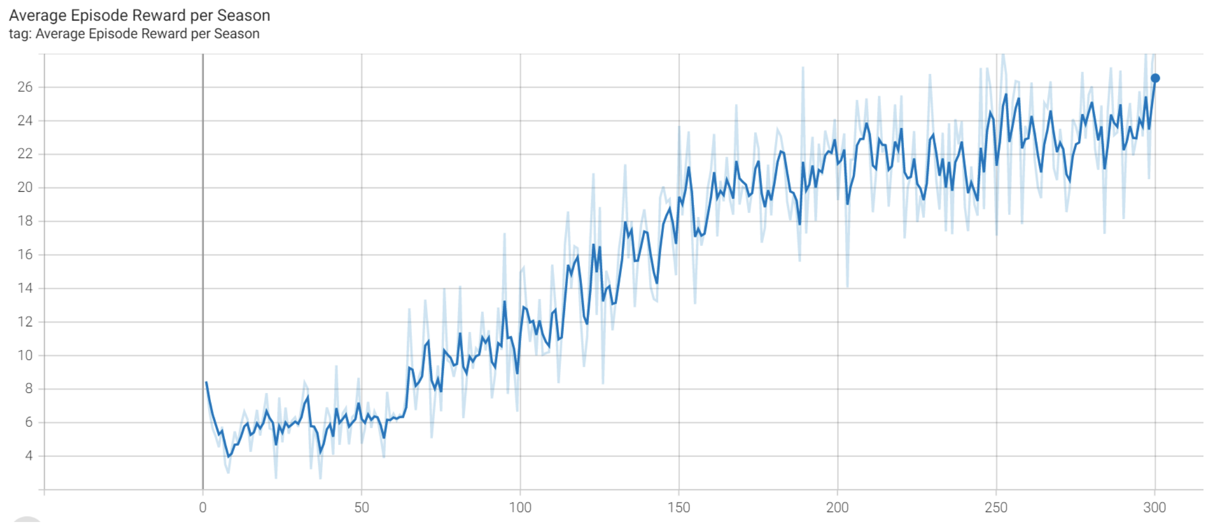

Fig 3. Average Episode Reward per Season vs nth Season

The reward curve learns faster from 60th to 200th episode and then the rate of learning slows down. The curve has reached a decent score of 26 around the end.

Training data and Test Simulation: Drive

The agent tries to avoid obstacle and change lanes way before it gets close to that obstacle. The agent is being extra cautious. It has learnt the right strategy to move to the right lane if the obstacle is on it's present lane, moving to the right lane increases the reward. As much as the agent stays right the reward gets higher. It has also learnt to move to the left lane if the obstacle is on it's present and right lane. It sometimes changes lanes unnecessarily where it needs to be idle and another drawback of the agent is that it inevitably crashes when it's near an obstacle.

Deep Q Learning

DQN collects a replay buffer of trajectory samples of size 5000, from which it randomly picks 64 or 128 samples to train on every 20 timesteps of samples collected. Linear and exponential epsilon decay strategies are used to pick random actions for exploration. Three variations of the trained results are as follows:

Variation 1:

Kinematics grid of nearest vehicles is used as observation and vehicle density is set to 1.5 which is a significant rise in complexity from 1.0. 3000 episodes had been run and exponential decay of epsilon is followed as exploration strategy. The epsilon falls down from 1 to 0.001, the decay parameter is set to 0.9975. A random number is generated at each timestep, if it's lower than the epsilon then a random action is chosen or otherwise the policy network provides the action to be executed in the simulator.

Epsilon in the nth episode = $1 \times (0.9975^n)$

Result:

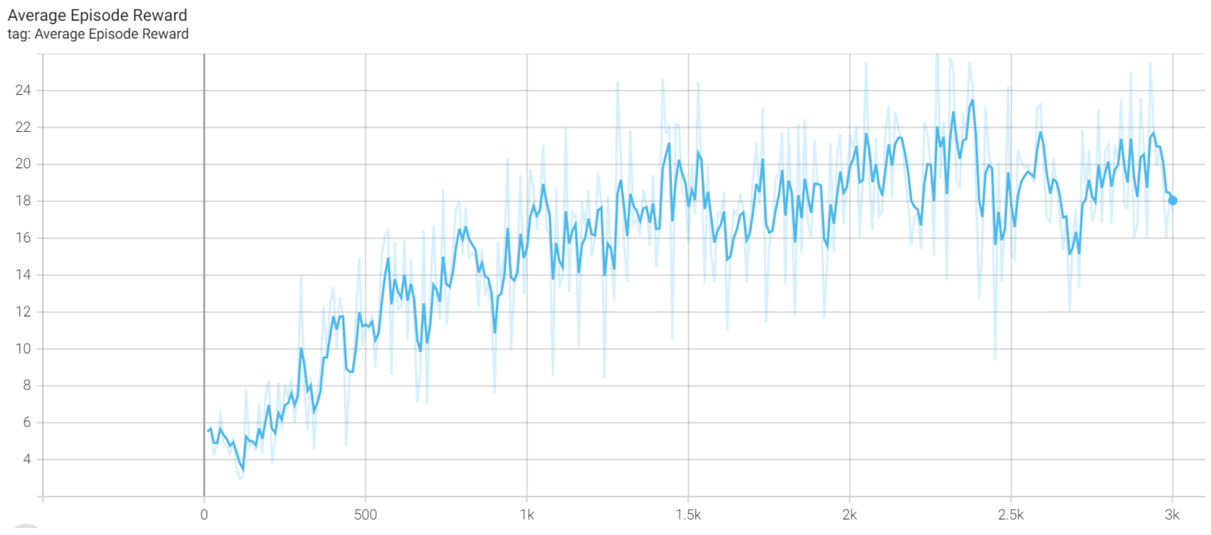

Fig 4. Average Reward of last 10 Episodes vs nth Episode

We can observe a learning curve which rises and then settles down and oscillates a little around the score of 19 to 20 per episode. Further there couldn't be any improvements after 3k episodes as the curve saturates.

Training data and Test Simulation: Drive

The agent avoids initial obstacles by changing lanes but it crashes or slows down immediately after to avoid obstacles rather than going faster and changing lanes to avoid them. It stays idle after a point without any changes in the lane. The agent struggles to identify the obstacle lane and takes the wrong decision.

Variation 2:

Kinematics grid of nearest vehicles is used as observation and vehicle density is set to 1.5. 10,000 episodes had been run and linear decay of epsilon is followed as exploration strategy. The epsilon is 1 at 1st episode and 0 at 10,000th episode.

Epsilon in the nth episode = n/10,000

Result:

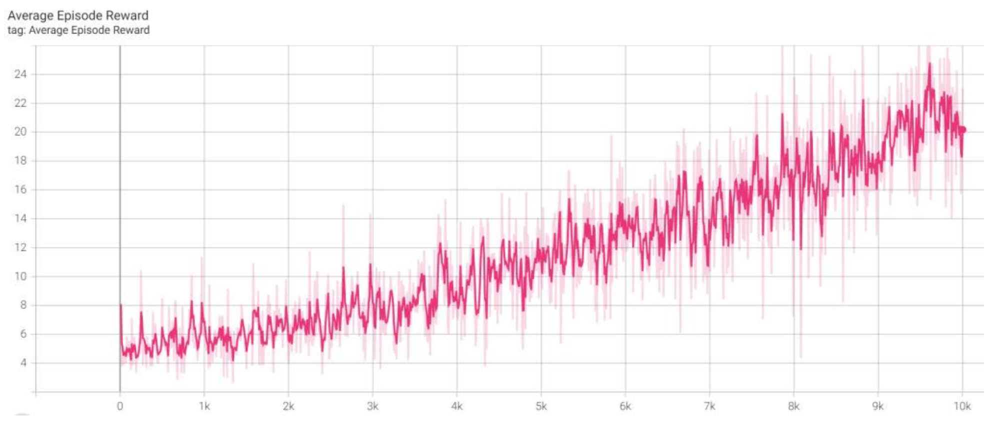

Fig 5. Average Reward of last 10 Episodes vs nth Episode

We could only observe the learning curve and no saturation curve, the rewards progressively increases till the end of the 10,000 episodes.

Training data and Test Simulation: [Drive]

The agent avoids most obstacles by changing lanes but in the later period when the episode approaches it's time limit it crashes directly straight into the rear end of the obstacle on the same lane. The agent turns late so that it inevitably crashes into the obstacle Infront of it. The agent struggles to identify the obstacle lane and takes the wrong decision.

Variation 3:

Occupancy grid is used as observation and vehicle density is set to 1.5. 3000 episodes had been run and linear decay of epsilon is followed as exploration strategy. The epsilon is 1 at 1st episode and 0 at 3,000th episode.

Epsilon in the nth episode = n/3,000

Result:

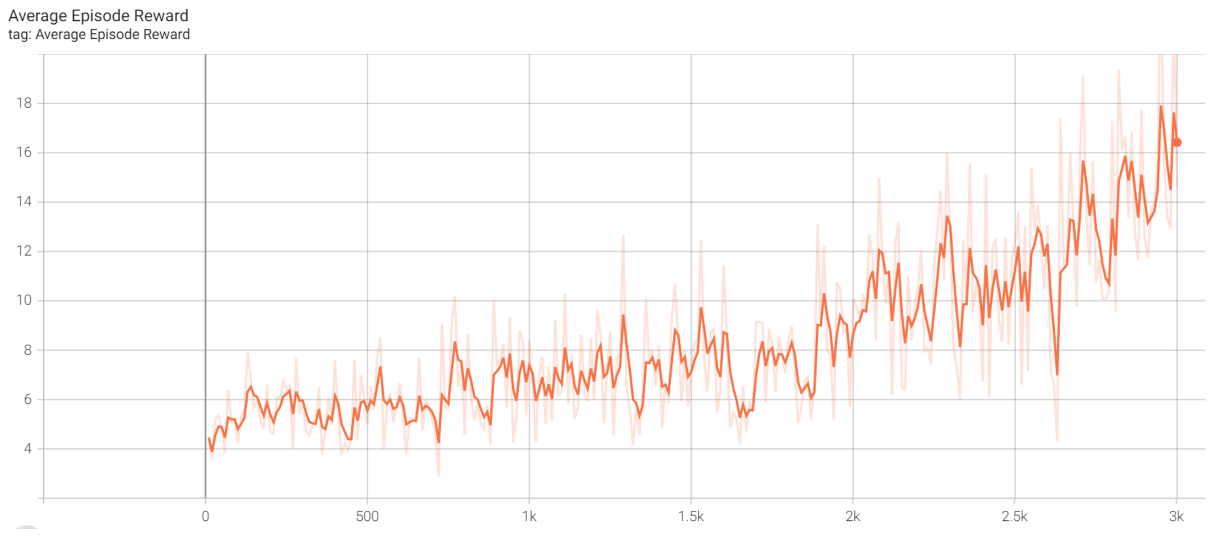

Fig 6. Average Reward of last 10 Episodes vs nth Episode

We could only observe the learning curve and no saturation curve, the rewards progressively increase till the end of the 3,000 episodes.

Training data and Test Simulation: [Drive]

The agent perfectly identifies the obstacle Infront of it and changes lane so close to the obstacle. The agent is very good at avoiding obstacles Infront of it but few times it misses the obstacles in the adjacent lanes and crashes into them when it moves to the next lane.

Conclusion

The best trained model simulation video. The linear epsilon decay variations never saturate and rises throughout all the episodes irrespective of the total number of episodes, whereas exponential decay and ppo saturates. PPO variation 2 gives us the best training result but during testing DQN variation 3 is robust in avoiding obstacles and being consistent in avoiding obstacles Infront of it. Occupancy grid observation gives us better testing results than kinematics grid observation because occupancy grid directly points out the obstacles relative to the agent. Both PPO and DQN seems to be able to solve this environment.

Code

https://github.com/thiyagutenysen/Highway

References

[1] https://github.com/eleurent/highway-env

[2] https://highway-env.readthedocs.io/en/latest/index.html