Episode – is length of the simulation, in our case we end an episode if there is a collision or if the episode is running for more than assigned threshold time.

Q learning – lets say we have some state space(basically every subtle change in our environment) and some action space. For a current state we need to perform some action from our action space to reach end goal, we usually perform the action which has highest Q-value.

$$q_*(s,a) = E\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right]$$

q*(s,a) is the Q value of a state(S) and an action(a)

R(t+1) is the reward that we get by performing action(a) on state(s)

S' is the new state that is generated by performing action(a) on state(S)

Max(q(s',a')) is basically the maximum q value over all the actions in our action space for the state(s').

Using this equation known as Bellman equation, for every episode our Q values get updated

Action space – in our case we have 3 actions, move forward, left and right..

Reward function – where we generate the reward depending on the state of the agent, whether it is collided or moving slow or fast. In our case we assigned rewards as -200 for collision, +10 for speed>50 and -10 for speed<50.

Epsilon – it decides about the next action that need to be taken whether we should explore or exploit. Explore is where we take some random action among the action space. Exploit is where we take the action that has the highest Q value. Initially we cannot exploit as there are no Q values, we have not trained our agent. So we start with epsilon = 1, which tends to explore (take random action), gradually after each episode we decay our epsilon value by 0.001, in this way after some number of episodes the agent exploits the environment and take action with highest Q value.

Step function – is where we perform a certain action to the state and assign the reward from the reward function. For eg. If action==0 we move forward

Action==1 turn left

Action==2 turn right, and assign reward from reward function discussed above. This function returns a some parameters that is used to generate a tuple for our replay memory(discussed down). The tuple consists of (current_state, action, reward, new_state, done)

Here new_state is the state that is generated by performing action on current_state and the corresponding reward for performing that action. Done is a Boolean to whether we end episode or not.

Replay memory – is a buffer of fixed size(let N) consists of the tuples that are generated for each frame in the episode used to train our neural network.

Why we need neural network? Actually in Q learning we need a table to know the max Q-value for a particular state to take that corresponding action. Here our state space is discrete we cannot store everything in a table, so we use neural network to predict the Q-values of a particular state.

Here we use two NN's one for predicting current_state Q-values (say policy network)

Other for predicting next_state Q-values (say Target network)

One not sufficient? No, because in a NN weights are updated by calculating the loss between target value and predicted value. Here our target value depends on next_state's Q-value as we seen in equation. If we used the same network to fit next_state values it tends to get close to it's next_state(precisely the next's next_state) and current_state values tends to go close to next_state, which makes the optimization like chasing its own tail and makes model unstable.. So we need two different networks.

Policy network gets updated whenever we get N length of tuples in our buffer, we fit the model with inputs(images) and target ouputs.

Target Network gets updated for every 10 updates of Policy network => if policy gets updated 10 times, target gets updated (basically assigning weights of policy).

The Flow:

- We initialize policy and target with some random weights and decides how many episodes to run.
- Episode loop starts, grabs the image from rgb camera attached to car (we are using two sensors rgb cam and a collision sensor), according to epsilon decides what action to be performed.
- Generates a state-action pair (current_state, action) here current_state is image from rgb and action is from previous step depends on the epsilon.
- Passes this to step function where the above discussed tuple is generated and the tuple is added to replay_buffer.
- In background as we discussed above the training of the NN will takes place and the weights of the NN gets updated…
- After the updation of weights with lot of episodes the NN gets somewhat familiar with the environment and generated good Q-values used for prediction.