Name: Jean-Pierre Richa
Matricola: 1779952
Report to Prof. Luca Iocchi

Sapienza University of Rome
AI 2b Reinforcement Learning Project Report
Deep Q-Learning for Self-Driving Cars

I. Introduction

Technological advances are helping to prepare the way for fully autonomous vehicles. Reinforcement learning, is a growing area of research and application. Within numerous applications for reinforcement learning, self-driving cars can do a great evolution if used for cars to learn the best way to execute the needed action to be taken by the car at each state. While reinforcement learning is used to learn an optimal policy that can be used by the car to better choose the next action, a combination of a neural network and Q learning in reinforcement learning, enhances the learning capabilities of the agent (the car in this case) by choosing the best action through the set of possible actions at each state, so it acts like the brain of the agent while applying the q learning and taking the best probability among all the possible actions. A focus on perception for autonomous vehicles is necessary and also enhances the learning, which in this case is received by using sensors.

II. Description of the problem

Using a car that will be navigating through a road that will be drawn on the map, each time making a different road and raising the level of difficulty, to be harder for the agent(car) to converge and to make sure that the agent is learning.
What will be drawn on the map(taken from kivy), is used as sand and each time the car goes through the sand, it will receive a bad reward, and will slow down, so the velocity of the car will become 1(very low).
The more time taken by the car to reach the goal the more it will get negative rewards.
The car travels with a velocity of 9, the velocity of 9 is used to make the car travel faster, and to make the learning faster by reducing the time needed by the car to execute each action, or traveling from the initial state to the goal state
If the car hits the edges of the map, it will receive a bad reward.
The car starts from an initial state which is the upper left corner of the map and should reach a goal state which is the bottom right side of the map.
When the car reaches the goal state, the goal state will become the initial state and the initial state will become the goal state.

The car has sensors that can perceive if there is sand, and/or it reached the edges

III. Formal model of the problem

1- Set of states (S):

The set of states is the set of all the possible states that the agent can be at, which is formed here by the initial state (is $\in$ S), goal state (gs $\in$ S), and the states in between where the agent is traveling inside the map. The initial state is defined by the 100 pixels on the upper left corner of the map, and the goal state is defined as the 100 pixels on the bottom right corner of the map, and vice versa. The current state of the car is defined by the number of pixels taken by the car inside the map, which is (10 x 5).

2- Set of actions (A):

The actions that can be taken by the agent are forward, left, or right. An action (a $\in$ A) is decided based on the current location of the agent, the distance between the agent and the goal, and if there is sand around the agent that can be perceived by the sensors around the car.

3- Transition function (δ):

The transition function is the transition from the current state to the next state through the application of an action (a $\in$ A).

$$s_{t+1} \sim T(a_t, s_t, .)$$

4- Reward function (r):

A reward function is given to the agent after it performs an action from a given state and result in a new state. There are 2 types of rewards: the immediate reward and the cumulative reward. The immediate reward is given to the agent when performing an action(left, right, or forward in this case) and resulting in a new state. Based on the distance between the agent and the goal in this new state, it is given a positive or a negative reward. The cumulative reward is the sum of the rewards accumulated during 1 episode which is in this case reaching the goal state from the initial state.

5- What is known to the agent?

The agent does not know the best trajectory to reach the goal state, it should find the path through applying actions in each state maximizing its reward function until it reaches the goal. The agent has sensors, so it will detect sands around it and try to avoid it after knowing that sand gives it a bad reward.
The agent knows its position in the map, and the set of states are fully observable, because we are using stochastic actions based on a probability distribution, so the state resulting from the execution of an action is not known before the execution happens, but once the action is executed, the resulting state is known by the agent.

6- What is the goal of the project?

It is difficult to pose autonomous driving as a supervised learning problem only due to strong the interactions with the environment including other vehicles, pedestrians, roadworks, etc... Here we are introducing a new model that is able to detect changes in the world and learn the best action to be taken at each state, and eventually the agent will be able to complete the challenge after having being trained enough to learn about the various obstacles that it will face.

IV. Solution Algorithm

The algorithm that I will be using is Deep Q-Learning, where I will make the deep neural network learn the Q-function, in order to decide the best action through a probability distribution among the 3 actions (forward, left, and right) which are the predictions that we get-

$$Q(s_{t_B}, a_{t_B})$$

- to be taken in each state. After that I get the target, -

$$r_{t_B} + \gamma \max_a(Q(a, s_{t_B+1}))$$

-compute the loss.

$$\text{Loss} = \frac{1}{2}\left(r_t + \gamma \max_a(Q(a, s_{t+1})) - Q(a_t, s_t)\right)^2 = \frac{1}{2}TD_t(a_t, s_t)^2$$

-The best action to be taken will be decided using the softmax function-

$$a_t \sim W_{s_t}(.) = \frac{\exp(Q(s_t, .))^\tau}{\sum_{a'} \exp(Q(s_t, a'))^\tau}, \text{ with } \tau \geq 0$$

-and the temperature parameter, which will enlarge the difference between the probabilities of the actions; This will make the agent more certain about the action that will be taken, since using the temperature parameter will tell the softmax function these are the probabilities and the higher the temperature is, the more the higher probability will increase with more difference with respect to the other probabilities.

V. Implementation

1- Data structures used:

I am using a dynamic list in order to save the transitions that will be provided to the agent in order to learn from past transitions. I am pushing the new transitions in the memory that will make sure that the number of transitions saved does not exceed the memory length, which is 100 in this

case. transition will consist of 4 elements the first element is " s " (the last state), the second element is " s' " (the next state), the third element is " a " (the last action), and the 4th element is " r " (the last reward).

2- Main implementation details:

The deep neural network consists of the input layer, which accepts an input of 7 variables, taken from the signals that are generated by the sensors around the car that can detect if there is sand near the car, and orientation of the car. It consists also of 3 hidden layers (3 hidden layers were chosen based on experiments), and an output layer that will provide the 3 q values that represent the probability distribution of the 3 actions that can be taken in each state. A temperature value is used to increase the difference between the probabilities of the q values, that will be given to the softmax function that will decide the best action to be taken in a given state. When 100 transitions are saved in the memory dynamic list, the agent will start using the transitions to learn what happened previously and learn how to get rid of the bad rewards that it took in the previous transitions. The oldest transition will be deleted each time a new one after they reach 100 is appended to the list, so we can keep the newest 100 transitions.

The more time taken by the car to reach the goal state, the more it will receive bad rewards, which based on experiments the best negative reward to be given to the agent while navigating is (-0.1), because the reward that will be given to the agent for reaching the goal is 1 and if we decrease the negative reward more, the final reward will not be good enough.
If the current distance between the agent and the goal state is less than the previous one, the agent will receive a positive reward of (0.5), which will say to the agent keep on going in this direction, because this way you are receiving a positive reward.
When the agent reaches the goal state, it will become the initial state (the bottom right corner of the map), and the old initial state will become the goal state (the upper left corner of the map).

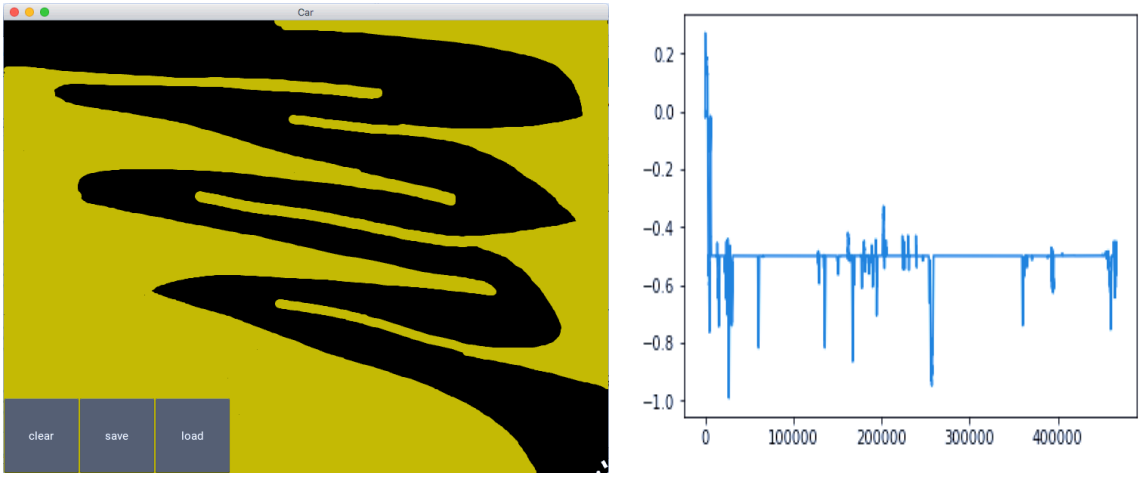
VI. Experimental evaluation

1- Comparative Solutions:

The map can be drawn by hand and I can dynamically change the world for the agent, so I don't have a static road where the agent can learn to travel it and we're done. On the contrary and as if it was in the real world, roads aren't the same, so the agent must learn how to cross different kind of roads, doing different kinds of maneuvers, dealing with crossroads, obstacles, etc…

In the 1st training, I tried to draw a complicated road for the agent to see if it's able to cross it with the help of the sensors, but as we will see in the following pictures, even with the sensors helping to keep it away from the sand, the agent wasn't able to learn a good policy and received a very bad reward.

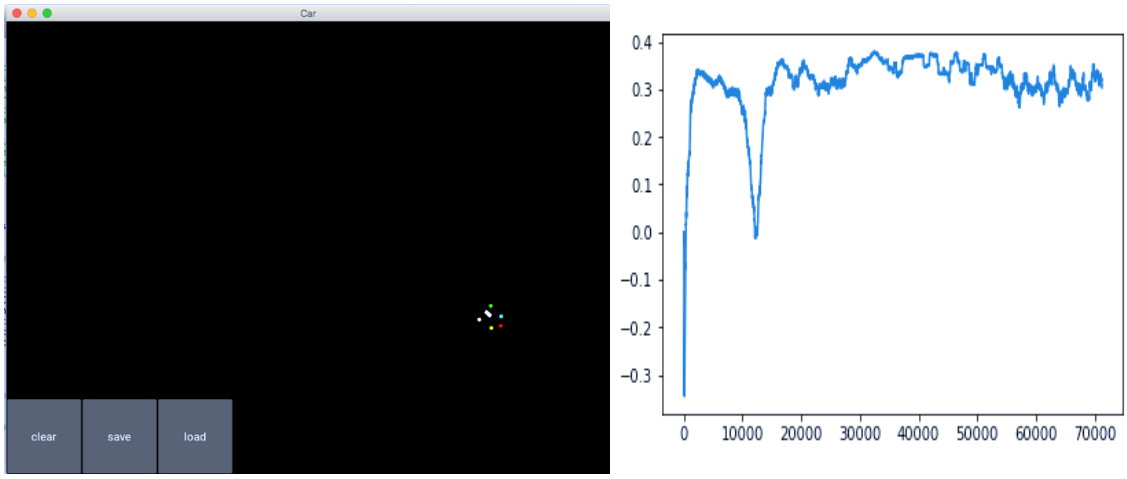
1st training



As we can see, the reward is under zero, even after 400,000 iterations, which Is not a good value for the learning agent. The value is between -1 and 1, but it can't hit 1, because the agent is getting a bad reward of -0.1 while traveling, so 0 and above is a good reward for the agent, the less it hits the sand, borders of the map, and the faster it crosses the map from the initial state to the goal state, the higher the reward will be.

2nd Training

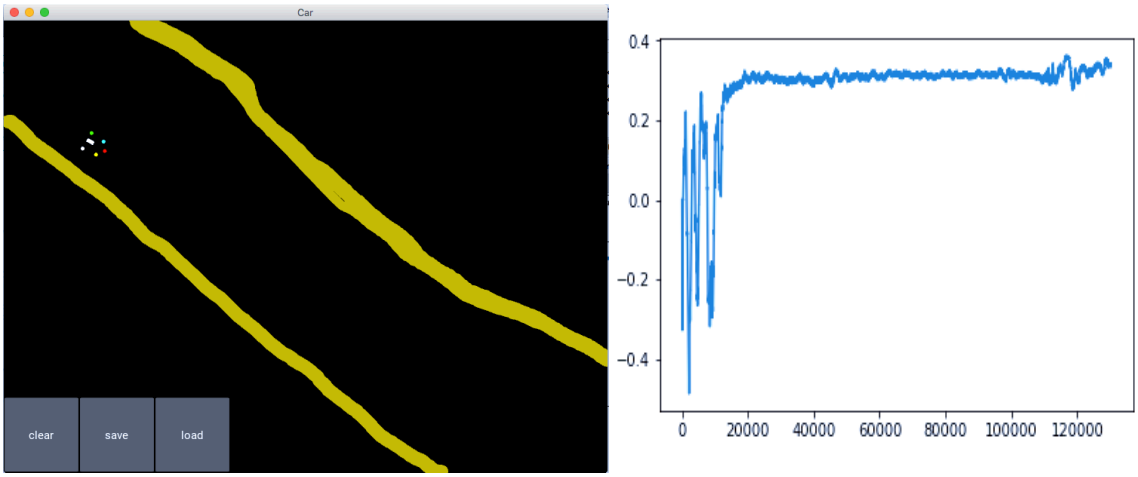
In the 2nd training attempt, I left the map empty and started to train the agent from the very basic step, to increase the difficulty later on.



Here we can see that the reward is pretty much the same (~0.4), after only 10,000 iterations, but I kept it training to make sure that the agent isn't going to do any mistake later. As we can see here, the best reward the agent can get is 0.4, because this is the best road (diagonally) crossed by the agent with the least possible time.

3rd Training

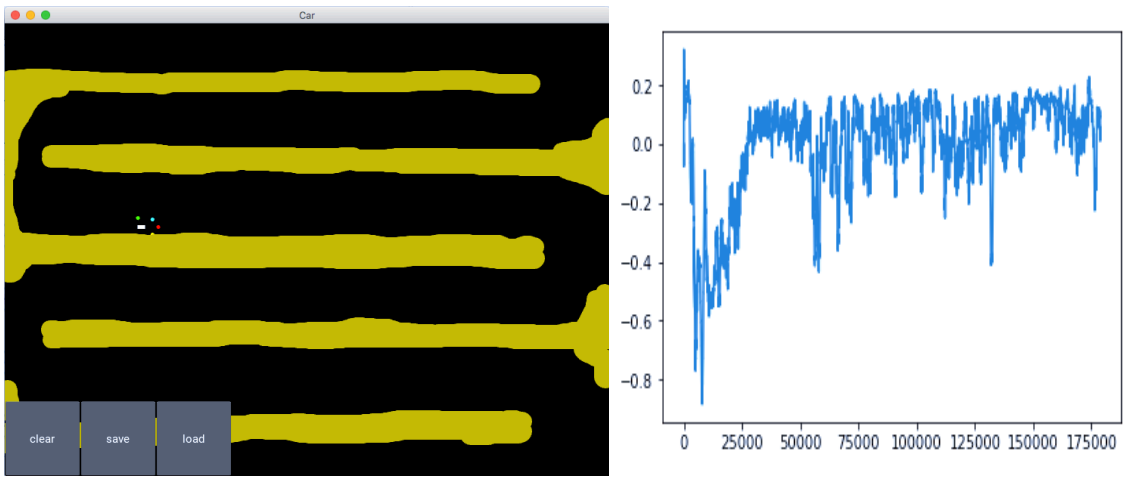
In the 3rd training attempt, I raised the difficulty a little bit, and loaded the last policy learned by the agent, as well as the weights updated by the agent from the previous empty map, because as shown in the picture I can save and load the last trained model, so I don't lose the progress.



In the beginning, as before, the agent struggled for a while to find its way in the updated road, but after 20,000 iterations, it was able to figure its way out.

4th Training

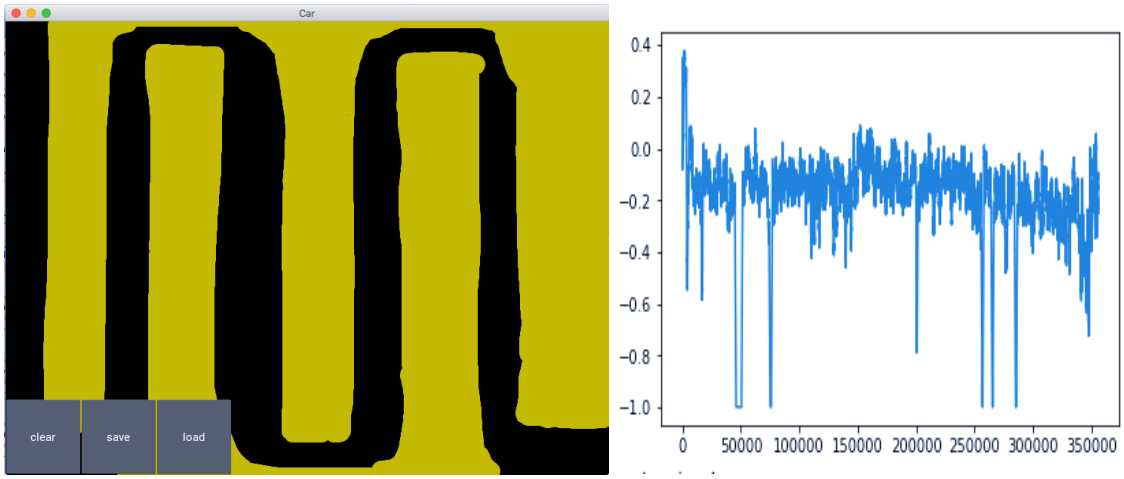
In the 4th training, I again raised the difficulty of the road, and loaded the last trained model. The save button overrides the last saved model, so the agent is adjusting its policy based on all of the previous experiences, and not on the current one only!



It is clear that this road is more difficult than the previous ones, and so the agent took more time to learn the policy function and have good weights for the neural network. After loading and using the previously trained model, we can see that the agent took nearly 175,000 iterations to converge with a pretty much stable reward. What is more difficult here is that the agent is getting a bad reward each time the distance from the goal increases, so drifting apart from the goal state decreases the reward more and more.

5th Training

In the 5th training, I changed the way the agent needs to travel, to make sure that the agent isn't learning to travel horizontally from the initial state to the goal.



As I already said before, here the challenge of getting a high reward is more difficult, because when we increase the distance to be traveled, the agent will be receiving more bad rewards, so after 350,000 iterations, we can see that we have a stable reward, so the agent was again able to learn a good policy.

VII. References

[1] https://kivy.org/
[2] http://webbut.unitbv.ro/Bulletin/Series%20I/2017/BULETIN%20I/Marina_L.pdf
[3] https://arxiv.org/pdf/1704.02532.pdf