Mai 2023

ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES

First year research project

Deep Reinforcement Learning for optimizing traffic using autonomous vehicles

By

Yentl Collin, *Pierre-Louis Franco, *Julien Galiano, *Dhia Garbaya

Supervisor :

Prof. Nadir Farhi

Advisor :

Prof. Amaury Hayat

* Equal contribution

CONTENTS

1 Key words and abstract

Key question : How does deep reinforcement learning algorithms allow for a safe and efficient way to implement autonomous vehicles on a highway section in order to enhance traffic fluidity ?

Key words : Deep Reinforcement Learning, autonomous vehicles, traffic fluidity, highway, optimisation, neural network, agents, optimal behavior

Abstract

Today's world is more crowded than ever, and the existing road infrastructures in most countries are not adapted to the number of users. Knowing that traffic congestion has an impact on the economy, the climate and the population's health, it is crucial to create solutions regarding this issue. Deep Reinforcement Learning enables an agent to learn how to behave through a trial-and-error process, using artificial neural networks. Such algorithms are particularly interesting in high dimensions problems and complex environments, having multiple interactions with the learning agent.
This paper discusses basic Deep Reinforcement Learning theory, such as Markov Decision's Processes and the basis of the exploration-exploitation dilemma. It also overlooks the different methods commonly used to build and train RL agents.
Relevance of DRL for the issue of traffic congestion is also evoked, as well as the modeling of road sections using the software SUMO. The approach taken regarding the subject is to introduce level-5 autonomous vehicles in traffic jam areas through freeway ramps, in order to form a file of vehicles traveling at a fixed velocity.
We explain how phantom traffic will mathematically occur in a natural way.
And we conclude with a survey of an open source RL highway environment and the re-establishment of results obtained by DQN and PPO.

2 Introduction

2.1 Presentation of the problem, and of the constraints

In November 2022 the world population reached 8 billion inhabitants and it is estimated to grow to 10 billion by 2050. With so many people on Earth, transport is and will be one of the sectors that needs to be improved and remodeled, in order to enable everyone to go from one place to another easily. Today one of the most popular means for these trips is cars and motorbikes, and with the growing population, there will be more and more people on the road.

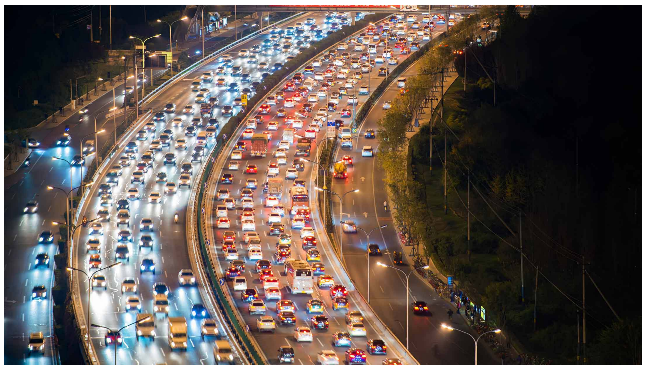


FIGURE 1 – Traffic jam in Beijing

Traffic has an impact on plenty of domains, be it economy, ecology or health and having an important number of people using their cars leads to several problems especially in an urban environment **[16]**. Indeed, traffic produces an important amount of carbon emissions (31 % of France's emissions were due to transport in 2019 **[17]**). Having more people on the road also means that the probability of having an accident will grow, and that traffic jams will occur more frequently. Knowing that traffic jams create carbon emissions and accidents, slow people in achieving their activities (and thus affect the economy) and cost more money in fuel, it is essential to prevent them.

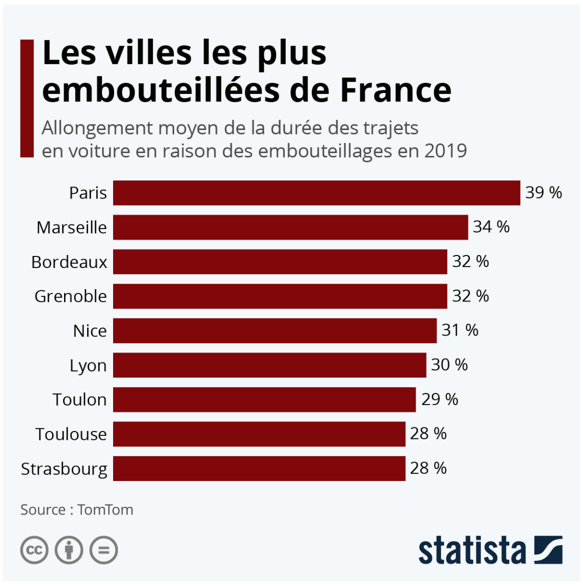


FIGURE 2 – Average lengthening of car trips due to traffic jam in 2019

As this graphic shows, many cities in France are impacted by the important number of vehicles on the road. Even in a city like Paris where public transport is well developed, the time needed to go from a point to another is 1.4 times longer than what it should be in a fluid traffic.

There are different solutions to this problematic. The first one is to build more roads but that is costly, time consuming, and it does not respect the necessity of reducing carbon emissions. A second option is to create efficient public transport means as it is already the case in most metropolises around the world. However, it is less effective in semi-rural areas where traffic jams also occur and public transport is much less developed. Another alternative is to promote carpooling especially knowing that in France, 80% of the people on the road are alone in their cars **[18]**. However, we will focus on a different method which is AI for traffic management.

2.2 Current state of research and industry

AI can be used in different ways in order to relieve urban traffic congestion. It can be used to regulate traffic lights **[2]** in order to keep traffic fluid and organized by adapting the signage in real time. Currently there are companies working on this problem, such as NoTraffic, but most of the traffic lights that are used are non-adaptive which means that the duration of the lights are fixed : it is obviously not optimal to control traffic fluidity.

In our case we will focus on the use of autonomous cars. There are different levels of automation (ranging from level 0 which is a vehicle with no type of automation to level 5 which is a vehicle fully autonomous that doesn't require any human control). By introducing such vehicles into traffic, one could regulate the speed of the vehicles behind them, which will allow the pack to reach an acceptable cruising speed instead of accelerating and braking every few seconds.

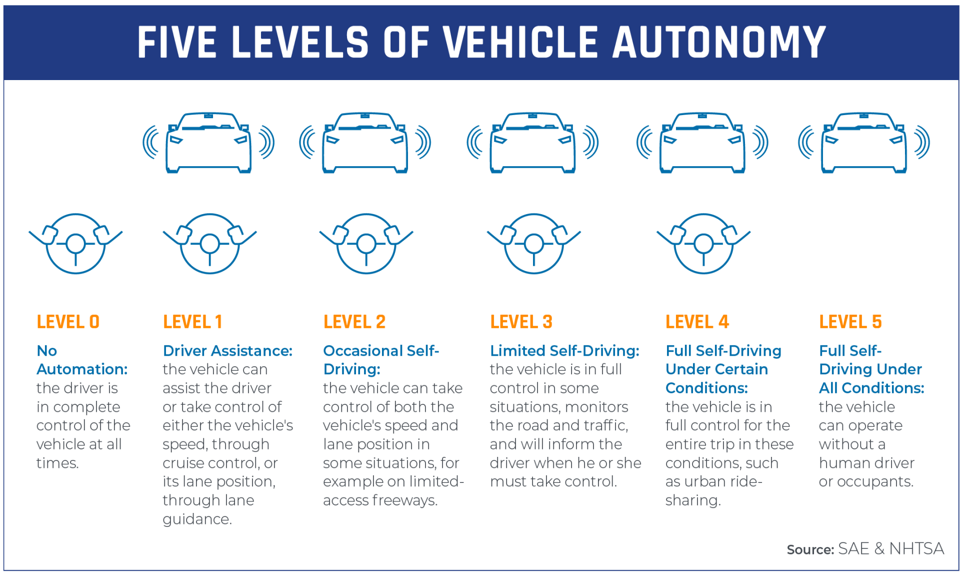

FIVE LEVELS OF VEHICLE AUTONOMY
LEVEL 0
No Automation: the driver is in complete control of the vehicle at all times.
LEVEL 1
Driver Assistance: the vehicle can assist the driver or take control of either the vehicle's speed, through cruise control, or its lane position, through lane guidance.
LEVEL 2
Occasional Self-Driving: the vehicle can take control of both the vehicle's speed and lane position in some situations, for example on limited-access freeways.
LEVEL 3
Limited Self-Driving: the vehicle is in full control in some situations, monitors the road and traffic, and will inform the driver when he or she must take control.
LEVEL 4
Full Self-Driving Under Certain Conditions: the vehicle is in full control for the entire trip in these conditions, such as urban ride-sharing.
LEVEL 5
Full Self-Driving Under All Conditions: the vehicle can operate without a human driver or occupants.
Source: SAE & NHTSA

FIGURE 3 – Five levels of vehicle autonomy

It is not yet legal in France to have an autonomous vehicle of level higher than 3 but the legislation will change as the number of level 5 autonomous vehicle built will grow.

3 Reinforcement learning

Traffic is a complex system, constantly evolving, with many actors at play. As such, focusing on a single model does not enable us to understand the different phenomena that occur. Indeed, we cannot optimize traffic management that way. Therefore, reinforcement learning is one of the promising approaches for our problem, as it is an approach that uses learning from interaction between the actor and the environment.

3.1 Introduction to RL

There exist three different approaches in machine learning : reinforcement learning, supervised learning and unsupervised learning. Supervised learning, which is currently the most popular method of learning, uses a set of example inputs provided by an external operator and generalizes this knowledge in order to act correctly in unknown situations. Unsupervised learning, on the other hand, does not use any given information but tries to find structure present in unlabeled sets of data. Finally, reinforcement learning is rather an optimization or optimal control approach. The learning agent is given a final goal and a reward function, and must act in order to maximize its reward function by trying different courses of actions in interaction with its environment.

In order to mathematically formalize decision-making problems, we use Markov Decision Processes (MDPs). In MDPs, at each time step t, the agent, which is both the learner and the actor, interacts with its environment. Each interaction leads to a new state and the agent earns a reward. This can be visualized by the following schematic taken from **[1]** :

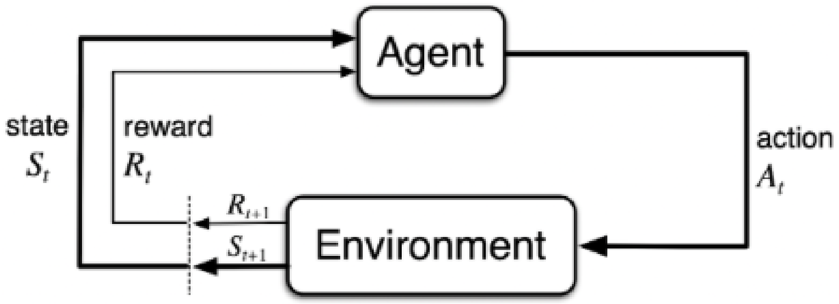


FIGURE 4 – The agent-environment interaction in a Markov Decision Process

In order to create an effective RL problem, it is necessary to clearly define what the goal is. We then introduce reward signals given at each action and the goal is translated into a maximization of rewards received by the agent.

At a timestep t, the agent, which is in state S_t, takes an action A_t which has an impact on the environment. The agent is then given a reward, with the environment having changed ; it is now in a state S_{t+1}. It is important to note that the agent must not maximize immediate rewards but long-term rewards. In order to do so, we introduce the notion of expected return as follows :

$$G_t = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}$$

with $\gamma \in [0, 1]$ called the discount rate. It gives different weights to rewards according to the time at which they are received. The more γ is close to 1, the more the agent will take into consideration future actions. On the opposite, having γ close to 0 means that the weight we give to the last rewards will decrease really quickly.

Let π be the mapping from states to probability of making an action. We call the map π the policy. We then express the reward value from making action a being in state s under policy π as :

$$q_\pi(s, a) = E[G_t | S_t = s, A_t = a]$$

By maximizing the reward function, the agent figures out the optimal policy π_* and we obtain the Bellman optimality equation :

$$q_*(s, a) = E[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a]$$

Solving the dynamic programming equation on the action-value function variable will permit to determine the optimal policy π for maximizing q_π. However, in reality even with a very thorough model containing both its properties and dynamics, it is hard to obtain an optimal policy thanks to this equation, especially in high dimensions problems. Therefore, some approximations need to be made.

3.2 Method to approach the optimal solution

In order to approximate realistic reward functions and optimal policies, Monte-Carlo methods are generally used, as they do not require prior knowledge of the environment and are based on experiences made by the agent. To approach an acceptable estimate of the reward function, an average of the rewards earned by going to each state must be made and the timeline must be divided into different episodes. The more states are explored, the more the estimate will be close to the expected value.

We begin by providing an initial arbitrary policy π_0 and the agent will execute different steps to obtain optimal values for policy and reward function. For each step, approximate reward function and policy are obtained. With a given policy, the agent works towards obtaining a more finely approximated reward function than at the precedent step and based on this new approximation, the policy is then improved. This cycle is called generalized policy iteration (GPI).

Mathematically this approximation of the policy can be translated as follows for each given state s :

$$\pi(s) = arg \max_a q(s, a)$$

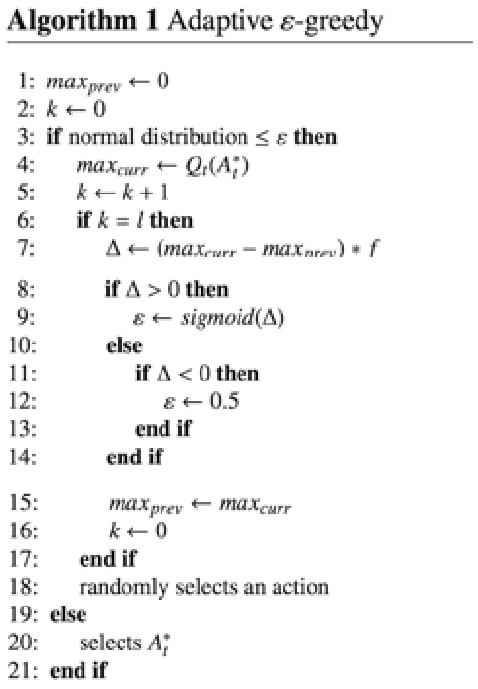
and for a step k $\in$ [1, N] :

$$q_{\pi_k}(s, \pi_{k+1}) = q_{\pi_k}(s, arg \max_a q_{\pi_k}(s, a)) = \max_a(q_{\pi_k}(s, a)) \geq q_{\pi_k}(s, \pi_k)$$

As such, each policy obtained is at least equal to the previous one. It shows that ultimately, both reward function and policy are converging towards the optimal ones.

But then comes another dilemma : the exploration-exploitation dilemma. First we need to define what exploitation and exploration are. We talk about exploitation when the agent selects an action that maximizes the current value function at each state in order to maximize its rewards. However, the value function being approached and not the optimal one yet, there are chances that it might not get the most reward in the long-term. Exploration is an approach that allows the agent to try actions that are not optimal for the current value function but it might lead to long-term benefit, since it gives the agent a better understanding of its environment **[14]**. When exploring, the agent is not working towards maximizing its reward function, but only exploiting might prevent him from making any new potentially better discoveries allowing to improve the policy in the end. Thus, finding an appropriate balance between exploration and exploitation is mandatory in order to reach a good approximation of the value function and a working algorithm. As such, a working balance must be found. In order to find this balance there exists different methods : undirected exploration methods and directed exploration methods.

The first type of method consists in giving each possible action a positive probability allowing the agent to browse a rather important part of the state-action space. A common method used for this type of exploration is the ϵ-greedy algorithm. We define probabilities for each possible action. We give a rather high probability of making an action that maximises the estimated reward function (exploitation) and a low probability ϵ of making a random action (exploration). However defining which ϵ is good is another dilemma, because at the beginning the agent has very few knowledge of its environment so it is more interesting to do random actions in order to gain knowledge and as time passes, the agent has a better understanding of its environment so exploitation is better. This approach is modeled by adaptive ϵ greedy algorithms **[9]** in which the value of ϵ is changed throughout time. We describe one of the adaptive ϵ-greedy algorithms, as they are used in section 5.3.2 during our analysis of a DRL training phase. We introduce two parameters l and f. The first one characterises, the length of the steps before changing the value of ϵ and the second parameter f is used to give a new value of ϵ.



Algorithm 1 Adaptive ε-greedy

```
$max_{prev} \leftarrow 0$
$k \leftarrow 0$
if normal distribution $\leq \varepsilon$ then
    $max_{curr} \leftarrow Q_t(A_t^*)$
    $k \leftarrow k + 1$
    if $k = l$ then
        $\Delta \leftarrow (max_{curr} - max_{prev}) * f$
        if $\Delta > 0$ then
            $\varepsilon \leftarrow sigmoid(\Delta)$
        else
            if $\Delta < 0$ then
                $\varepsilon \leftarrow 0.5$
            end if
        end if
        $max_{prev} \leftarrow max_{curr}$
        $k \leftarrow 0$
    end if
    randomly selects an action
else
    selects $A_t^*$
end if
```

FIGURE 5 – ϵ-greedy algorithm taken from **[9]**

However, there are shortcomings to undirected exploration methods. The most important one being that the number of steps they require grow exponentially with the size of the state space. Directed exploration methods rely on past experiences in order to do well-organized exploration. A common strategy used is optimism in the face of uncertainty. At first the algorithm assumes that all actions result in an arbitrary reward (that may be higher than what they will be in reality) then the actor tries an action and if the reward is less than expected then the algorithm will lower the action estimated value, and will focus on another action. It will also compare it with the arbitrary reward. Otherwise, the algorithm will continue with the selected action **[15]**.

3.3 Focus on Deep Reinforcement Learning (DRL)

Deep Reinforcement Learning is a subfield of RL combining both RL and Deep Learning(DL). DL enables the agent to learn the action-value function using an artificial neural network, instead of using tabular methods. The neural networks which are formed by different nodal layers, most of them being hidden and one being the input layer while another is the output layer. Each node is linked to another neuron with a weight and a threshold associated to the path. For each layer, the information is transmitted to the next layer only if the value obtained in the first layer is higher than the transition threshold. Each node can be considered as a linear regression model taking into account the input data, the weight and the threshold. The weight allows to formalize the importance of data. Deep learning is especially helpful when dealing with complex and high-dimensional input data.

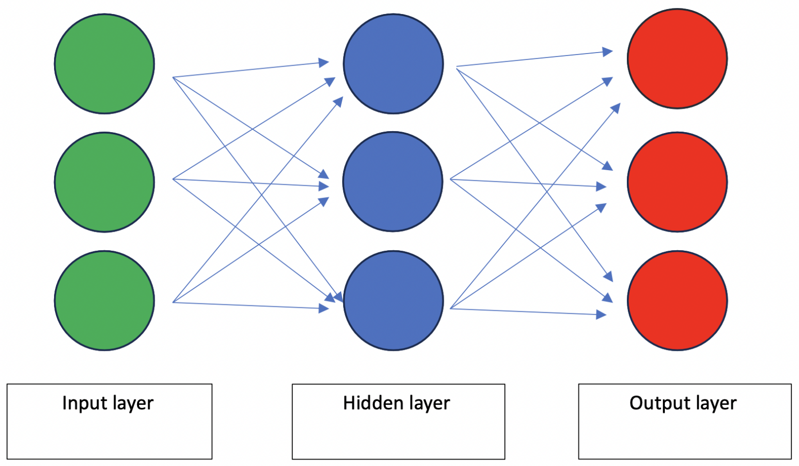


FIGURE 6 – Simplified diagram of an artificial neural network

In many complex environments, the states set S can have a high-dimensional, making MDPs impossible to be solved by traditional RL algorithms, and can only be solved when embedding neural networks in the algorithm. For instance, the policy π can be represented as a neural network. **[19]**.

4 DRL for traffic fluidity

Deep reinforcement learning can be used to solve optimization problems, so that we would be able to implement fully autonomous vehicles into the traffic, in order to enhance it.

4.1 Relevance of DRL for traffic fluidity

Deep learning and reinforcement learning are indeed particularly relevant approaches for studying traffic, compared to other available methods. Indeed, traffic data can be very massive and complex, which can make it difficult to analyze. Traditional techniques, such as decision trees or linear regressions, may be insufficient to accurately model the complex relationships between different traffic variables.

However, models based on Deep Reinforcement Learning are able to automatically extract patterns from these data, and thus determine even non-linear relationships between different traffic variables. Moreover, these models can handle raw data, without the need to manually extract features or explanatory variables, which can avoid the introduction of potential biases.

One can then ask what to consider to determine qualitatively if the traffic is fluid. We then consider the fundamental diagram, that gives a simple definition of traffic fluidity and congestion :

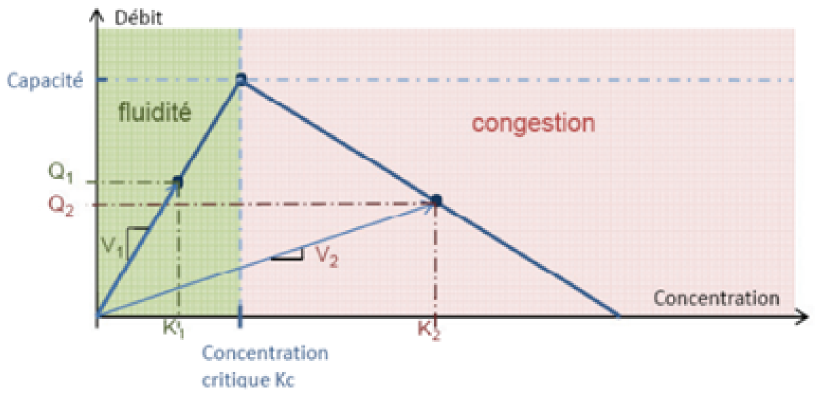
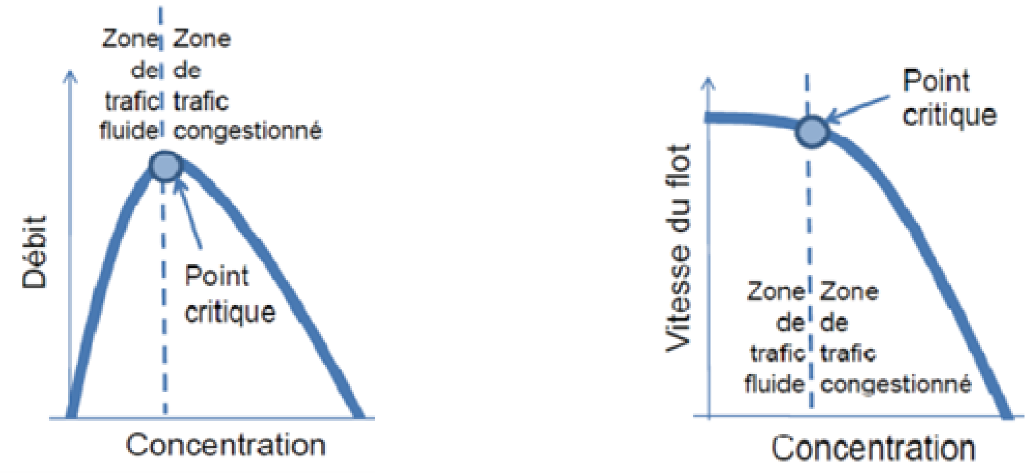


FIGURE 7 – Fundamental Diagram of Traffic Flow taken from **[20]**

When it comes to traffic, the flow depends on the density of vehicles (that must be below a critical threshold). Congestion, on the other hand, occurs when the arrival of a new vehicle deteriorates traffic for other users. The left-hand side of the diagram represents free-flowing traffic, characterized by a constant speed equivalent to the authorized limit on the road. This corresponds to a stable and desirable level of service. However, the right-hand side of the diagram represents congested traffic, which is characterized by a decrease in speed when the car-density is high. Indeed, the higher the density of vehicles is, the more interactions between vehicles there are, resulting in a decrease of the flow on the road. The whole Fundamental Diagram of Traffic Flow is a concave curve, and the critical density separates the two domains explained above.

With the introduction of autonomous cars, our goal is to maximize the level of service provided to users along the road. The aim of autonomous cars is to keep the concentration of vehicles on each segment below the critical concentration, or at least as close as possible to the critical value. This approach allows us to obtain a first value function to optimize traffic management.



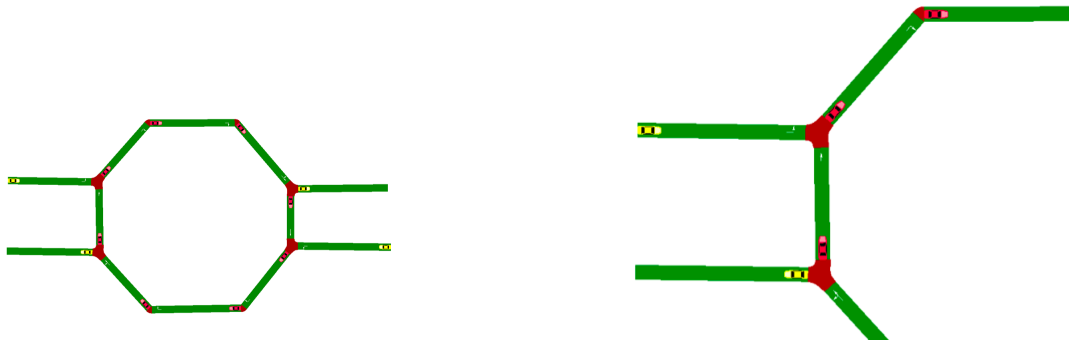
(a) Flow as a function of car concentration (b) Speed as a function of car concentration

FIGURE 8 – Difference between flow and speed curves taken from **[20]**

4.2 A possible modeling

Our first idea is to consider a road section with several entries and exits. The optimization is to know when the autonomous vehicles should enter and at what speed they should go to optimize traffic. The main difficulty is to create a flow of incoming vehicles on the road section, because vehicles that go beyond the last portion of the road are not taken into account anymore.

That is why we decided to focus on a more reasonable situation : a circle-shape road, as the one described in **[13]** (with the figure 8 in section 4.2). This time we only have to consider a fixed number of vehicles, which will indefinitely stay on the circular road. To create the simulations, we use the software Simulation of Urban MObility (SUMO).



(a) The whole modeling (b) Zoomed modeling

FIGURE 9 – Our circular road modeling on SUMO

This SUMO modeling is the illustration of the situation that we just described : the green sections are the road on which the cars are on. The first element is the octagon in the middle. Each section of the first element contains a single lane, and they are all oriented in the same direction (clockwise here). The red cars that are on it represent the flow that we want to enhance. The vehicles are not able to leave the octagon. In order to enhance traffic, we added two entries and two exits on the sides : these roads enable the autonomous cars (represented in yellow) to enter on the main road or to exit it. The decision to enter (or not) and to exit (or not) has to be found using DRL : we hoped to find a behavior for the autonomous cars so that the traffic is always optimized.

Therefore the parameters that we hoped to optimize are : when an autonomous vehicle has to enter the circular road, at what speed it should go so that the traffic could be improved, and when it would have to leave the circular section. We can of course fix a maximum speed for the autonomous vehicle. In general, a single autonomous vehicle can already enhance traffic. Indeed, its action can be decomposed into two phases. During the first phase, the autonomous vehicle slows down to stabilize traffic : people driving normal cars will stop accelerating and braking every few seconds. Then comes the second phase : the autonomous vehicle slowly speeds up again, to reach its cruise speed. This slow acceleration does not deteriorates the traffic flow, as it is done little by little.

5 Observations and analysis

At first, we wanted to code everything from scratch, as mentioned in the mid-project report. Shortly after that, we decided that it would be more profitable for us to analyze already existing results about DRL for traffic rather than coding without obtaining any result. Indeed, after a discussion with our referent on this project, Nadir Farhi, we realized that it would take too long to do for us, and that we would not be able to use it in the end. Indeed, we have spent time trying DRL on smaller and simpler examples and we did not manage to obtain conclusive results by ourselves, so we chose analysis instead.

However, our tests to create neural networks enabled us to understand better the way it works, and how to use libraries like PyTorch or TensorFlow. Indeed, even if we do not have any results to show that are obtained with our own code, we learned a lot about trial and error while coding : identifying the part of the code that does not work, looking up online when we have errors while compiling, looking up online the documention of libraries, and so on.

5.1 Source and Content

In this part, we're going to be more concrete by exploiting an open-source DRL for self-driving cars GitHub repository **[12]**. The latter contains several Python scripts that implement various DRL algorithms such as Deep Q-Networks (DQN), Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) to train agents in a simulated highway environment.
The HighwayEnv is an environment, or rather a collection of environments, that help realize a simulation of a highway with multiple lanes, where the agents (autonomous vehicles) have to navigate through traffic and reach their destination, while avoiding collisions and maintaining a safe distance from other vehicles.

What is interesting about this repository is that it provides several pre-trained models that can be used to evaluate the performance of the trained agents. The evaluation metrics include the average speed of the agents, the average number of collisions, and the average distance maintained from other vehicles.

5.2 Analysed Packages

To analyze the results, we have run some pre-trained models and observed the performance metrics.
Before getting to the analysis of these training results, it is important to note that the performance of the agents heavily depends on the hyperparameters used during training, such as the learning rate, discount factor, and exploration rate. Therefore, it is recommended to experiment with different hyperparameters to obtain the best results.

In what follows, we will make observations and offer interpretations of the training results of two implementations 'sb3-highway-dqn.ipynb' and 'train-PPO.py'. We will finally come back on some details about this complex environment by discussing cars pre-defined behaviors.

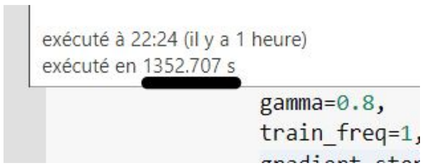
5.3 Hyperparameters

5.3.1 Sb3-Highway-DQN

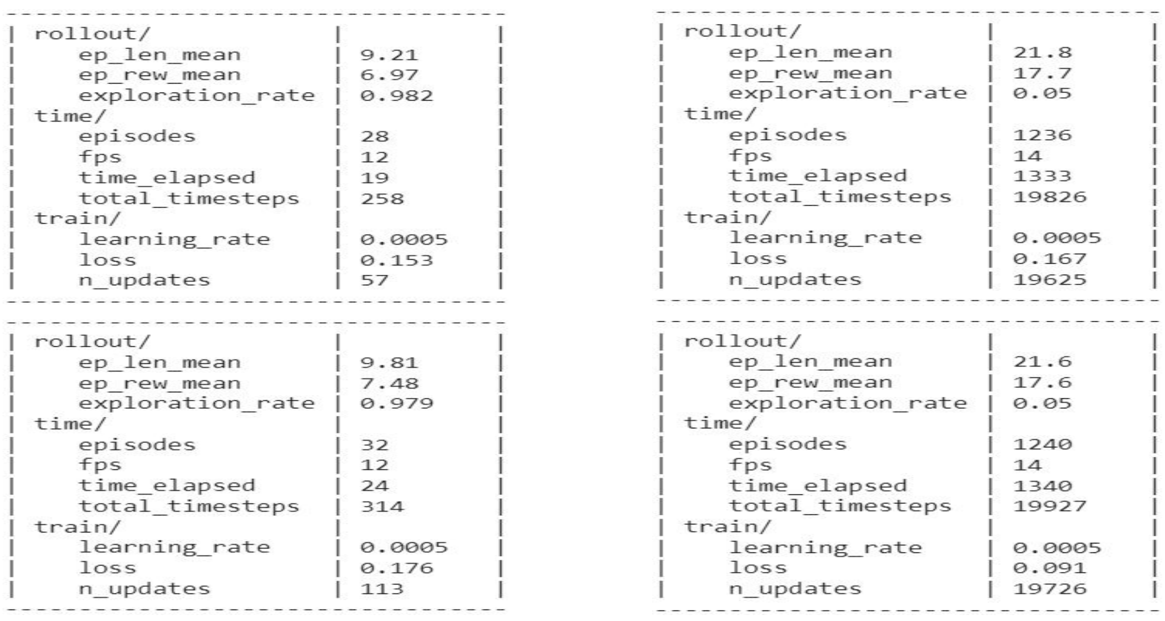
First, let's take a look at the 'sb3-Highway-Dqn' script. This script is used to train a Deep Q-Network (DQN) agent. Some key hyperparameters used in this script are :
— learning-rate : In this script, the default value is 0.0005.
— gamma : The gamma parameter is the discount factor that determines how much weight to give to future rewards versus immediate rewards. In this script, the default value is 0.8.
— train-freq : The train-freq parameter determines how often to update the neural network. In this notebook, the train-freq is set to 4.
— exploration-fraction : This parameter determines the fraction of the total number of training steps to use for exploration. In this notebook, the exploration-fraction is set to 0.7.
— buffer-size : The buffer-size parameter determines the size of the replay buffer used in the DQN algorithm. In this script, the default value is 15000.

```
model = DQN('MlpPolicy', 'highway-fast-v0',
                policy_kwargs=dict(net_arch=[256, 256]),
                learning_rate=5e-4,
                buffer_size=15000,
                learning_starts=200,
                batch_size=32,
                gamma=0.8,
                train_freq=1,
                gradient_steps=1,
                target_update_interval=50,
                exploration_fraction=0.7,
                verbose=1,
                tensorboard_log='highway_dqn/')
model.learn(int(2e4))
```

- The training process took between 22 and 23 minutes. And this after about 20,000 iterations during which our parameters evolved significantly.


exécuté à 22:24 (il y a 1 heure)
exécuté en 1352.707 s

```
gamma=0.8,
train_freq=1,
```



```
---------------------------------------------
| rollout/                  |               |
|    ep_len_mean            | 9.21          |
|    ep_rew_mean            | 6.97          |
|    exploration_rate       | 0.982         |
| time/                     |               |
|    episodes               | 28            |
|    fps                    | 12            |
|    time_elapsed           | 19            |
|    total_timesteps        | 258           |
| train/                    |               |
|    learning_rate          | 0.0005        |
|    loss                   | 0.153         |
|    n_updates              | 57            |
---------------------------------------------
---------------------------------------------
| rollout/                  |               |
|    ep_len_mean            | 9.81          |
|    ep_rew_mean            | 7.48          |
|    exploration_rate       | 0.979         |
| time/                     |               |
|    episodes               | 32            |
|    fps                    | 12            |
|    time_elapsed           | 24            |
|    total_timesteps        | 314           |
| train/                    |               |
|    learning_rate          | 0.0005        |
|    loss                   | 0.176         |
|    n_updates              | 113           |
---------------------------------------------
```

(a) First steps

```
---------------------------------------------
| rollout/                  |               |
|    ep_len_mean            | 21.8          |
|    ep_rew_mean            | 17.7          |
|    exploration_rate       | 0.05          |
| time/                     |               |
|    episodes               | 1236          |
|    fps                    | 14            |
|    time_elapsed           | 1333          |
|    total_timesteps        | 19826         |
| train/                    |               |
|    learning_rate          | 0.0005        |
|    loss                   | 0.167         |
|    n_updates              | 19625         |
---------------------------------------------
---------------------------------------------
| rollout/                  |               |
|    ep_len_mean            | 21.6          |
|    ep_rew_mean            | 17.6          |
|    exploration_rate       | 0.05          |
| time/                     |               |
|    episodes               | 1240          |
|    fps                    | 14            |
|    time_elapsed           | 1340          |
|    total_timesteps        | 19927         |
| train/                    |               |
|    learning_rate          | 0.0005        |
|    loss                   | 0.091         |
|    n_updates              | 19726         |
---------------------------------------------
```

(b) Last steps

FIGURE 10 – Evolution of parameters between the beginning and the end of the experimentation

- **Interpretation :** We note that while the learning rate remained in our case constant, the exploration one decreased drastically. Indeed, As the agent gains more experience and becomes more proficient in the task, it is less necessary to take random actions. Instead, the agent can exploit its existing knowledge to make more informed decisions.

5.3.2 train-PPO.py

We can also train a Proximal Policy Optimization (PPO) agent.
PPO is a policy-gradient algorithm introduced by OpenAI in 2017 to simplify TRPO.
Some hyperparameters used for PPO are :

— epsilon-decay : The epsilon-decay parameter determines how quickly the agent's exploration rate (epsilon) decreases over time. In this script, the default value is 0.0003.

— ent-coef : The ent-coef parameter is a coefficient that determines how much weight to give to the entropy of the policy. The default value is 0.01.

— vf-coef : The vf-coef parameter is a coefficient that determines how much weight to give to the value function in the PPO algorithm. In this script, the default value is 0.5.

To optimize the performance of these agents, one needs to carefully tune these hyperparameters, using techniques such as grid search or random search.

5.4 Results

- The result of the training process is then visualised in the form of a short episode (GIF) created thanks to the libraries record-videos and show-videos imported from 'utils'.
- The behavior is quite satisfactory since the agent (autonomous vehicle) manages to navigate the environment with remarkable efficiency and safety.

- **Script above** : We can see that at the beginning of the GIF, that the agent starts in the rightmost lane of the highway and maintains a relatively slow speed. As the agent gains experience and learns more about the environment, we can see that it starts to drive more aggressively, changing lanes frequently and driving at higher speeds.

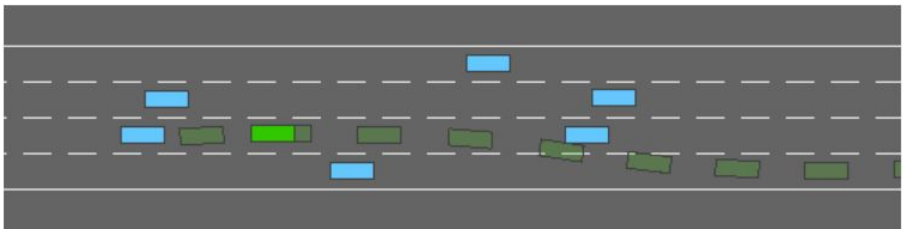


FIGURE 11 – Screenshot from an episode

One interesting behavior we can observe is the way it changes lanes to avoid slower-moving vehicles. When a slower vehicle appears in front of the agent, the agent accelerates and changes lanes to pass the slower vehicle, and then decelerates and changes back to the original lane. This behavior is a common one in human driving and is a good indication that the agent has learned a reasonable strategy for dealing with slower traffic.

Another behavior we can observe is the way the agent maintains a safe distance from other vehicles. The agent keeps a reasonable following distance from other vehicles and adjusts its speed appropriately to avoid collisions. This behavior is critical for safe driving and is a good indication that the agent has learned to prioritize safety in its decision-making, which is exactly what was expected from him.

Finally, we may note a tendency to get to the right side (lane). Indeed, this is predictable as driving on the right side of the road is rewarded in this case.

Overall, the agent's behavior in the GIF is quite impressive, and it appears to be driving in a safe and efficient manner. However, it is important to note that this is just a single example, and the performance of the agent may vary depending on the specific environment and task. It is always important to thoroughly test and evaluate trained agents before deploying them in real-world applications.

5.5 Pre-defined behaviours

Among the python packages we can find a script called 'behavior.py'.
This script defines the behavior of vehicles in the HighwayEnv environment. Here are some key parts of 'behavior.py' :

- class BehaviorModel : This is the base class for all behavior models. It defines the interface for behavior models, which includes the methods 'act' and 'decide'.
- class IDM : This is a subclass of BehaviorModel and implements the Intelligent Driver Model (IDM) for vehicle behavior. IDM is a car-following model that is widely used in traffic simulation and control. It computes the acceleration of a vehicle based on the distance to the leading vehicle, the relative velocity to the leading vehicle, and the desired velocity of the vehicle (cf. 4.1).

- N.B : These behavior models can be seen as a way to provide a starting point for the agent and to cover a range of scenarios that the agent may encounter during training or testing. While the ultimate goal is for the agent to learn an optimal behavior policy, the behavior models can help the agent to converge faster and to perform better in the long run.
In addition, some defined parameters are very useful for the RL process as they help define the reward for example. And let's not forget that these predefined behaviors are above all those that will govern the actions of other human-driven cars.

6 Conclusion

All in all, this project enabled us to discover Reinforcement Learning, even if we did not actually manage to code our environment from scratch, we had the opportunity to learn about implementations.

The first part of the project was mainly about reading articles and reports on this subject to familiarize ourselves with the global theme. We also tried to test small simulations, like the cart-pole simulation, a classic example that DQN or PPO get to solve easily.
You can see for instance github/dhia680/DRL/cartpole.

In the second part of the project, the analysis of the HighwayEnv repository enabled us to witness the importance of the different parameters of our problem and the particular sensitivity of DQN. Which turns out to be a well known weakness of DQN.
We also understood the stabilization techniques such as considering a policy network and a target network, momented updates of the network and so on.
In the end, we succeeded in visualizing simulations of an autonomous vehicle (single agent) on a highway section with multiple lanes, after a training phase of less than 1h.

If we had more time, it would have been interesting to try and analyze other algorithms and environments **[12]**. Indeed, our work was mainly exploratory, with an exploitation and analysis of existing RL initiatives.
But this was a great opportunity to get an early introduction to RL and deep learning in general. And to work a problem that we all face but never thought it could be approached this (fascinating) way.

We would like finally to thank Prof. Nadir Farhi, for his time and devotion to the project, who managed to guide us throughout the whole semester. We also thank Prof. Amaury Hayat, from Cermics, who shared insights from control theory and CIRCLES project he took part in, provided us with documentation and guided us to some implementations.

7 Bibliography

[1] : R. Sutton and A. Barto, “Reinforcement learning : An introduction (second edition),” 2018. [Online]. Available : http ://incompleteideas.net/book/RLbook2020.pdf

[2] : R. Ducrocq, N. Farhi, "Deep Reinforcement Q-Learning for Intelligent Traffic Signal Control with Partial Detection", IFSTTAR, F-77454, 2021. [Online].
Available : https ://arxiv.org/pdf/2109.14337.pdf

[3] : B. Ravi Kiran, I. Sobh, V. Talpaert, P. Mannion, A. Al Sallab, S. Yogamani, P. Pérez “Deep Reinforcement Learning for Autonomous Driving : A Survey”, 2021. [Online].
Available : https ://arxiv.org/pdf/2002.00444.pdf

[4] : E. Vinitsky, N. Lichtlé, K. Parvate, A. Bayen “Optimizing Mixed Traffic Flow with decentralized autonomous vehicles and Multi-Agent Reinforcement Learning”, 2022. [Online].
Available : https ://arxiv.org/pdf/2011.00120.pdf

[5] : R. Ducrocq, DQN-ITSCwPD, https ://github.com/romainducrocq/DQN-ITSCwPD

[6] : A. Dos Santos Mignon, A. da Rocha, “An adaptive implementation of epsilon-Greedy in reinforcement learning” Procedia Computer Science, 109, 1146-1151, 2017.

[7] : S. Cui, B. Seibold, R. Stern, D. “Work Stabilizing Traffic Flow via a Single Autonomous Vehicle : Possibilities and Limitations”, 2017.

[8] : A. Hayat, B. Piccoli, S. Xiang “Stability of multi-population traffic flows”, 2022.

[9] : M. Tokic, "Adaptive ϵ-greedy exploration in reinforcement learning based on value differences". In : Annual Conference on Artificial Intelligence. Springer, Berlin, Heidelberg, 2010. p. 203-210.

[10] : M. Stevens and C. Yeh, “Reinforcement learning for traffic optimiza- tion,” 2016. [Online].
Available : https ://cs229.stanford.edu/proj2016spr/report/047.pdf

[11] : E. Walraven, M. Spaan, B. Bakker, "Traffic flow optimization : A reinforcement learning approach", 2016.

[12] : @mischighway-env, Leurent, Edouard, "An Environment for Autonomous Driving Decision-Making", 2018, https ://github.com/eleurent/highway-env

[13] : C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, A. Bayen, "Flow : Architecture and Benchmarking for Reinforcement Learning in Traffic Control," CoRR, vol. abs/1710.05465, 2017. [Online]. Available : https ://arxiv.org/abs/1710.05465

[14] : S. Ishii, W. Yoshida and J. Yoshimoto, "Control of exploitation-exploration metaparameter in reinforcement learning", 2002.

[15] : T. A. Mann, Y. Choe, "Directed Exploration in Reinforcement Learning with Transferred Knowledge", 2012.

[16] : S. Faye, "Contrôle et gestion du trafic routier urbain par un réseau de capteurs sans fil", 2014.

[17] : Commissariat général au développement durable, "Les émissions de gaz à effet de serre du secteur des transports",2021, Available : https ://www.notre-environnement.gouv.fr/themes/climat/les-emissions-de-gaz-a-effet-de-serre-et-l-empreinte-carbone-ressources/article/les-emissions-de-gaz-a-effet-de-serre-du-secteur-des-transports .

[18] : Capital, "Plus de huit Français sur dix roulent seuls en voiture", 2022, Available : https ://www.capital.fr/auto/plus-de-huit-francais-sur-dix-roulent-seuls-en-voiture-1446591

[19] : V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau, "An Introduction to Deep Reinforcement Learning", 2018.

[20] : Buisson, Christine, et Jean-Baptiste Lesort, "Comprendre le trafic routier : méthodes et calculs", Éd. du Certu, 2010.