PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH


ABDERRAHMANE MIRA UNIVERSITY OF BÉJAIA

FACULTY OF EXACT SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

MASTERS THESIS

OPTION : ARTIFICIAL INTELLIGENCE

---

# Proposition of an Enhanced Deep Reinforcement Learning Algorithm for Managing Traffic

---

*Presented by:*
Amine HAMOUCHE

*Examining Committee:*

*Supervisor:* Dr. BOULAHROUZ Djamila

*Examiner 1:* Mme BOUKERRAM Samira

*Examiner 2:* Dr. Adel Karima

**Academic Year: 2021 / 2022**

# Declaration of Authorship

I, Amine HAMOUCHE, declare that my thesis titled, “Proposition of an Enhanced Deep Reinforcement Learning Algorithm for Managing Traffic” and all the work presented in it are my own. I confirm that:

- This work was done wholly on this year at my University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- Where the thesis is based on work done by myself, i am making it clear that i didn’t get the help from any of my classmates, except from my supervisor.

ABDERRAHMANE MIRA UNIVERSITY OF BÉJAIA

# *Abstract*

Faculty Of Exact Sciences
Department of Computer Science

Masters in artificial intelligence

**Proposition of an Enhanced Deep Reinforcement Learning Algorithm for Managing Traffic**

by Amine HAMOUCHE

Due to the rapid growth of the population as well as the quality of life, reliance and the use of roads is increasing drastically which leads to an inevitable increase in traffic. As a result, many traffic control systems have been in operation to regulate traffic but still need improvements to be more suitable especially in our country. Indeed, these systems are designed to work on predefined patterns which do not always correspond to the real and particular cases of our road traffic.

In order to tackle this challenge we propose in this work an upgrade approach of an already existing traffic monitoring model based on Deep Q-Learning to traffic lights control with an agent simulator using the framework SUMO. Our main objective is to minimize the average waiting time of the vehicles at the intersection. To this end we propose many improvements to the algorithm, by adding some important layers to the convolutional neural network and adapting the values of some crucial hyper parameters.

The obtained simulation results have shown that our approach performs better than the existing algorithm in terms of mean waiting time.

**KEYWORDS** : Traffic lights Control Systems, Deep Reinforcement Learning, Convolutional Neural Network.

# *Acknowledgements*

I would like to thank my supervisor, Dr.Boulahrouz Djamila for guiding me through this work as well as all the efforts and the time she put in to prepare me for presenting this thesis.

She offered me a great help in the preliminary work in order to understand the basics of the algorithm we used. I would also like to acknowledge my university for their participation and engagement in my study.

I would also like to earnestly acknowledge the sincere efforts and valuable time given by my examining committee. Mme BOUKERRAM Samira and Dr. Adel Karima.

A debt of gratitude is also owed to all the teachers of my university's curriculum, for giving such valuable lessons and some important learning experiences.

# Contents

# List of Figures

# List of Abbreviations

| | |
|---|---|
| **RL** | **R**einforcement **L**earning |
| **ML** | **M**achine **L**earning |
| **DL** | **D**eep **L**earning |
| **DRL** | **D**eep **R**einforcement **L**earning |
| **QL** | **Q**uality **L**earning |
| **TSC** | **T**raffic **L**ight **C**ontrol |

# General Introduction

Traditional traffic light control systems, which are still commonly utilized in metropolitan areas, have proven their limits and deficiencies in the management of actual traffic network. This is why, in recent years, traffic management systems have been the subject of extensive research.

One of the biggest challenges facing these systems is managing large amounts of vehicles at an intersection so that the average waiting time of these vehicles is acceptable. Indeed, the number of vehicles is rapidly increasing which leads to a substantial increase in waiting time at an intersection.

Several techniques have been proposed in literature [1],[2],[3],[4], however reinforcement learning as an enhanced machine learning technique, seems to be the most interesting approach to develop smart traffic control systems.

In this thesis, we tackle the problem of managing traffic light control systems at an intersection aiming to minimize the average waiting time of the vehicles. We consider that the total amount of time vehicles spent waiting at a red light is measured as a total waiting time. Our model is based on the work of [1], which we propose to improve for better performance in terms of the average waiting time for red lights at intersections. The main objective is to allow the large public benefit from this time saving and to respond more efficiently to eventual emergencies. In addition to that, such efficient of traffic light systems might be very helpful in the fight against global warming.

The authors of [1], which is the basis of our model, have proposed a relatively easy solution using a deep reinforcement learning algorithm (Q-Learning) which alters the timing of traffic light control system in response to current traffic demand. Although the results they have obtained are interesting, they can yet be enhanced by adjusting the model's key parameters. Our work is within this context. We propose to improve the model of [1] by adding some important layers (Batch normalization, Dropout) to the convolutional neural network and adapting the values of some crucial hyper parameters (Learning rate, mini-batch size, Epoch, Activation function, Adam Optimizer).

We conduct several simulations of our model by testing various parameters values in order to get the optimal ones. The obtained simulation results have proven that our model has significantly outperformed the algorithm of [1] in terms of the average waiting time.

Our thesis is structured as four chapters :

- **Chapter 1** : Covered traffic signal systems in general, from conventional to intelligent systems. We have also reviewed some methods for managing traffic on the roads and the significance of traffic engineering. Finally, we discussed how traffic light systems affect CO2 emissions and how a better overall design would result in less emissions.
- **Chapter 2** : Summarized the significant breakthroughs that convolutional neural networks have made in various fields of application is provided in this chapter. After reviewing several fundamental concepts related to reinforcement learning, Q-learning, which combines it with deep learning into Deep Q-learning algorithm, was presented.
- **Chapter 3** : Discussed the method used to frame the problem so that the operating principle could be better understood. We then rapidly moved on to the structure the algorithm's creators had chosen. In addition, we showed the many improvements we did, from changing some important parameters to adding beneficial layers to neural network.
- **Chapter 4** : Addressed the many tools used to integrate the improvements did and also the software used to simulate the overall environment. After that, we showed using graphs the results of how our enhanced algorithm. we demonstrated that our proposed approach made an overall increase in performance by reducing the average waiting time of the vehicles at the intersection.

# Chapter 1

# A state of the art of traffic light systems

In this chapter, we will try to formulate a concise state of the art concerning traffic light systems.

## 1.1 Traffic lights systems

Improving traffic flow in cities is one of the top priorities, with it road safety can also be drastically improved. it is in that perspective that intelligent traffic lights have been designed, a traffic light is a device sometimes multiple devices connected that allow the regulation of traffic between roads, vehicles and pedestrians at different places we will be seeing it at an intersection. Depending on the color displayed by the traffic light, they allow whether or not users will be allowed to cross the intersection. it is a system of management by priorities and based of three different color : red, orange and green that not only concerns vehicles but also the pedestrians.

FIGURE 1.1: Traffic light sequence

### 1.1.1 Traditional traffic light systems

These traffic light systems are mainly based on fixed time delays with sometimes few options for the pedestrians and often they work with a defined sequence of lights, Green, Orange, Red, Green, etc.. These traffic lights use a

timer to change the lights at a fixed interval, the timer depends on the historical traffic data of the said intersection. The fundamental goal of this strategy is to accomplish a broad target such as lowering average delay, increasing the capacity of the network, etc [5] created the original model and provided the groundwork for fixed-time traffic light control systems, the downside is that even a minor disruption like a construction site or traffic collision can drastically change the traffic flow thus compromising its effectiveness.

### 1.1.2 Dynamic traffic light systems

Later, visual systems' application became widespread with it and with the limitation of fixed-time traffic light control, dynamic traffic lights systems appeared they employ a detector that can interact with the traffic light control system to provide them with information about the flow of traffic. They will alter timing when a junction is congested to improve traffic flow.

### 1.1.3 Smart traffic light systems

Since they first appeared traffic light control systems have contributed to fewer accidents and more optimized traffic in urban areas, the growth in the number of vehicles these recent years is an important factor to consider. Recently, a number of projects and research endeavors have been devoted to the creation of fresh approaches that permit management of traffic light control systems while accounting for various situations. A new traffic suggestion made by [3] combines object detection with evolutionary algorithms and machine learning to reduce waiting time for vehicles as well as pedestrians. As only one traffic light at any intersection needs to be fitted with smart technology, smart or intelligent traffic lights provide excellent value for money, the issue is that for the system to function optimally, vehicles must also use smart technologies like GPS, driver assistance systems that keep an eye on pedestrians, traffic and speed.

FIGURE 1.2: Smart traffic light systems

#### The significance of smart traffic light systems

One can use a straightforward example to understand the true value of smart traffic lights, if a vehicle goes too slowly or too fast the smart traffic system

will identify and will decide which way to resolve the problem in order to avoid any accidents. Given how quickly technology is developing, one can only speculate about how clever and practical these systems will eventually be.

### 1.1.4 Theoretical tools for road traffic management

Dynamic models are frequently employed in the literature and rely on theoretical techniques, sometimes by approximating reality and other times by ignoring physical principles (technology used, layout). In traffic management systems, a number of theoretical tools that are widely discussed in the literature and used as the foundation for several models.

#### Using fuzzy logic

Fuzzy logic makes it possible to set up degrees in the verification of a condition, allowing one to expand their options beyond strictly binary ones. this theory is employed by some authors to deal with the problem of traffic light management and allow to simplify, which modifies standard mathematical optimization techniques frequently hefty. Additionally, [6] which uses fuzzy logic to decide the length of a traffic light based on number of cars on the road (for instance less than five vehicles per minute grant 10 second green light). It is relatively a simple theory to apply , the drawbacks are also important since the settings are empirical and no theory can show that they are stable.

#### Neural Networks

Neural networks use learning experience and are modeled after how biological neurons function, numerous authors have researched this plan for road traffic. It is also possible to think that fuzzy logic and other algorithms like genetic algorithms as enhancements to neural networks.

#### Queuing theory

Road traffic management falls under the domain of probability and lends itself particularly well to the queuing theory [7], since it allows the most effective use of queues. In queuing theory, when cars (called clients) can get a green light at a junction, a line is immediately created (server). It is quite simple to also calculate quantities with this theory, such as the average number of vehicles waiting $Nbrv$, in service, the average waiting time $T$ and the average staying time $TS$ in the system. It is conventional to use Kendall's notation while drafting a queue based system, the three symbols $a/s/C$ are used to represent the system. Whereas $a$ denotes the law of probability guiding the circumstances of arrival times and $s$ the service duration, which is typically exponential ($M$ or $G$). for its part $C$ represents the quantity of servers. This theory's drawback when it comes to managing an intersection it necessarily requires numerous measurement points.

## 1.2 Traffic Engineering

Traffic engineering covers all the telecommunications-related means used to manage and control how traffic is distributed throughout a network by dealing with the design of roads, streets in addition to those areas where they intersect with other motorized and non-motorized modes of transportation.

### 1.2.1 Objectives of traffic engineering

The authors of [8] give a relatively helpful understanding of the main goal of traffic engineering. Simply put it is to provide a secure and safe system for vehicular transportation. in addition, lowering trip time and enhancing the comfort with technology being the only constraint to it.

### 1.2.2 Importance of traffic engineering

Unnecessary and excessive traffic signals at once can be dangerous, one of the biggest barriers to implementing sound traffic engineering principles is the widespread misconception that installing a traffic signal will automatically solve any traffic issues. The engineer counts the amount of traffic, studies crash statistics, speeds data, looks at the state of the roads, conducts research and examines what other experts are doing and the outcomes they have attained.

## 1.3 Impact of traffic management on CO2 emissions

The design of mechanisms that increase the effectiveness of road use, such as real-time traffic information systems and collaborative routing systems, is being done concurrently with the attempt to create more efficiently and environmentally friendly cars. The authors of [9] believe that a greater attention should be paid to the mechanics of how each kilometer is driven, that includes the choice of route and that is because they stated that vehicles having a shorter wait time at an intersection would mean a lesser release in CO2. They maintain their idea that having careful step not only on the overall safety but also on the environment.

## 1.4 Conclusion

We have discussed traffic light systems in general in this chapter. From traditional to smart traffic systems, we then reviewed some tools for road traffic management and the importance of traffic engineering. We finally concluded with the impact of traffic light systems on CO2 emissions and how a better design overall would impact less CO2 emissions.

# Chapter 2

# Analyzing neural networks and understanding reinforcement learning

## 2.1 Introduction

Nowadays artificial intelligence techniques are no longer a mystical treasure kept hidden from the public but it is a growing tool that are penetrating all kinds of different fields, transportation is no exception the aim is to increase the mobility of people as well as their goods by making the traffic road healthier by trying to choose the optimal traffic duration thus minimising it. Most traffic lights systems work on a similar fashion, they cycle through the green, yellow and red colors to ensure a congruous flow of traffic in the intersection.
This kind of systems are well suited for busy areas that have a consistent volume of traffic but they quickly become outdated in areas that do not expect a lot of drivers to be around thus the necessity to rely on an "intelligent" control system, the approach of RL algorithm chosen is a promising one in fact it can learn to improve the traffic by simple observation done by an agent followed by algorithmic adjustments for optimization.

## 2.2 Analyzing artificial neural networks

Computational models called artificial neural networks are modeled after the nervous system of living beings [10], they can be characterised as a collection of processing units represented by artificial neurons. McCullosh and Pitts proposed in 1943 a simple neuron model, which incorporates the key components of a biological neural network connectivity and is still one of the most used model(see figure 2.1).

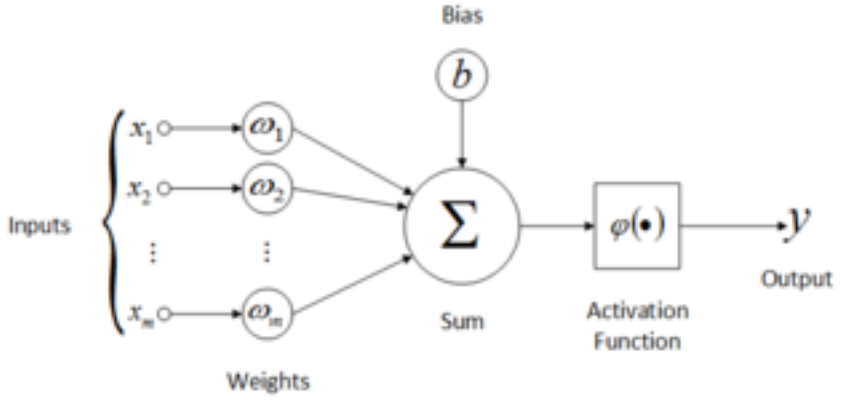


FIGURE 2.1: Artificial Neuron Network model

- *x1,x2,...,xn* are the samples and signal originating from the outside environment.
- *w1,w2,...,wn* are the factors that determine how each input variable is weighted, allowing for the assessment of its significance.
- $\sum$ sums all the weighted signals to produce an activation.
- *b* the bias in artificial neural nets helps to replicate the behavior of real, human neurons.
- $\phi$ is an activation function in a neural network describes how a node or nodes in a layer of the network translate the weighted sum of the input into an output.
- *y* consists of the final value the neuron produces after receiving a specific set of input signals.

## 2.3 Analyzing convolutional neural networks

Authors of [11] Give a pretty helpful understanding of the architecture within the deep learning domain over conventional machine learning. In this work, we will use convolutional neural networks combined with Deep Q learning. Convolutional neural networks (see figure 2.2), or CNNs for short, are a subset and one of the most effective image categorization models at the moment, their mode of operation is quite straightforward : the user has to enter as input an image in the form of a matrix of pixels. The matrix has 3 dimensions; two dimensions for a gray scale image and a third one to represent the colors, the design of the convolutional neural network is based on the classical multi layers perceptron model which has only one classification part but CNNs have an added convolutional component.

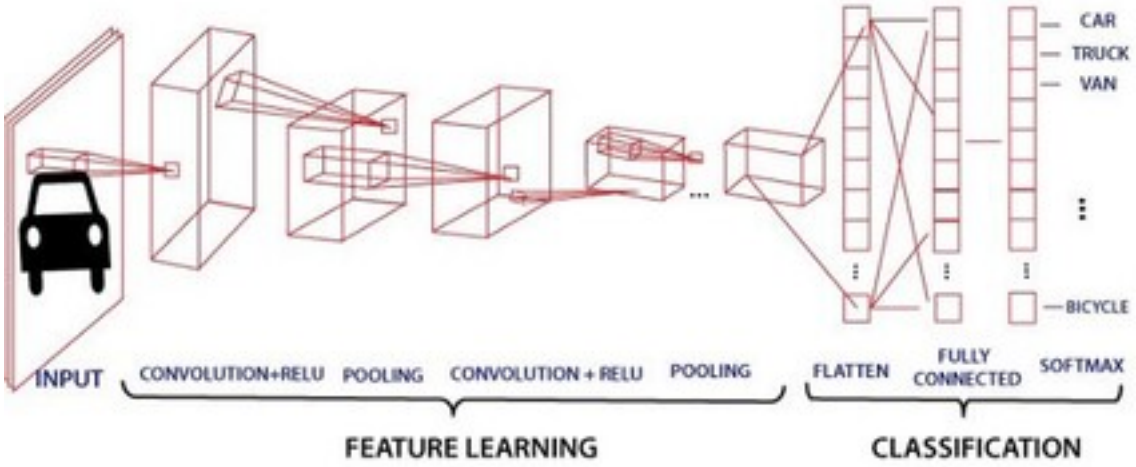


FIGURE 2.2: Convolutional neural network

### Convolution part

The goal of the convolutional part is to lower each image's starting size while extracting attributes unique to each one, this is the part in which if an image of a dog was given as input it will begin to recognise the ears and legs as an example (see figure 2.3) . The input image is processed through multiple filters (or kernels) each of which produces a new image called a convolutional map that will finally be concatenated into a vector. We use an activation function to pass the output as non-linear usually it's the ReLu function.

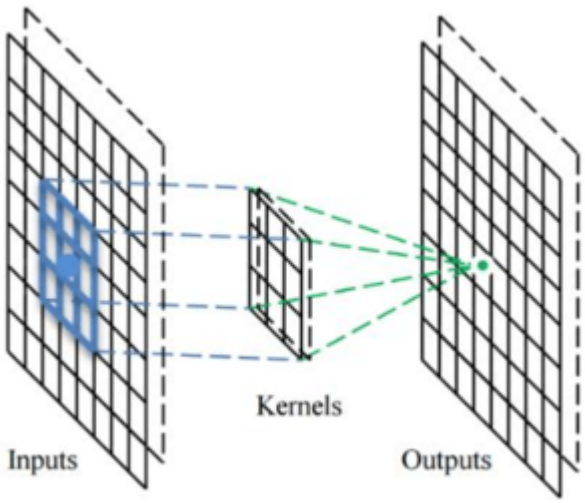


FIGURE 2.3: Convolution process visualisation

#### Classification part

The convolution part seen earlier happens in the hidden layers of the neural network, the connected layers will help us assign a probability *y* for the object on the image being what the process predicts it is (see figure 2.4).

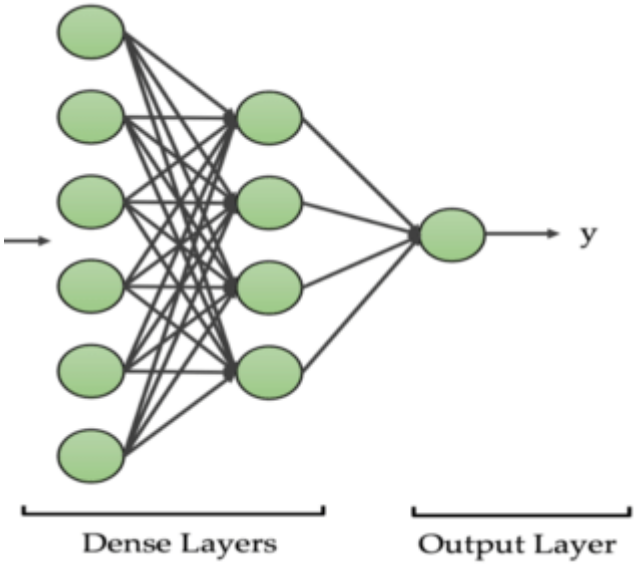


FIGURE 2.4: Classification process visualisation

### 2.3.1 Various CNN architectures

Many CNN architectures were and are still developed, some well known ones are : LeNet architecture, AlexNet an improved LeNet, GoogleNet architecture.

#### LeNet architecture

The most classic Convolutional Neural Network, known as LeNet, was created by Yann LeCun et al in 1990 [12]. The LeNet architecture that was designed to scan zip codes, numbers, etc. is one of the most efficient architectures.

#### AlexNet architecture

The earliest well-known CNN architecture is AlexNet, which was created by Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton [13], popularized convolutional neural networks in computer vision. AlexNet's architecture was the most deep and profound of them all, LeNet's architecture was comparable to that of AlexNet. Stacking convolution layers as opposed to the changing convolution and pooling layers used in LeNet.

#### GoogleNet architecture

This CNN architecture developed by [14] from Google was unique in that it drastically reduced the amount of parameters in the Inception v-1 network module, it is also stated that its performance was close to a human level performance.

### 2.3.2 Applications of convolutional neural networks

In this section, some fields and domains where convolutional neural networks were used are presented.

#### Image Classification

Convolutional neural networks have been used for a long time in the field of image categorization. Compared to alternative approaches CNNs perform better on huge datasets. In 2012, large-scale picture categorization achieves its breakthrough. The AlexNet was developed by Krizhevsky et al. [13], who also had the top results in the ILSVRC 2012. Following AlexNet's success, a number of works have significantly increased classification accuracy by either decreasing filter size or increasing network depth.

#### Text Detection and Recognition

Long-term research has been done on the problem of text recognition in images. Traditionally, The main emphasis is on optical character recognition (OCR). Text recognition is the core function of OCR technology on pictures in visually restricted settings.

#### Object Detection

A long-standing and significant issue in computer vision is object detection. The challenges often center on finding the locations of objects in pictures or video frames. CNNs have been used for detection and localization since the 1990s. However, given the absence of The development of CNN-based object detection is delayed due to a lack of training data and processing resources.

#### Action Recognition

One of the difficult challenges in computer vision is action recognition, which involves analyzing human subjects' behavior and categorizing their activities based on their visual appearance and motion dynamics. Action analysis in still photos and in video are the two main categories into which this issue may be separated.

## 2.4 Reinforcement Learning

The RL approach (see figure 2.5) used differs drastically from the supervised and unsupervised machine learning techniques [15], it learns based on successive experiences, what is best to be done in order to get the optimal solution. Recent studies show that RL approaches have shown a great success in different fields like finance, communication and scheduling. Trial and error is the foundation of learning.
This approach is applied for any problems where the decision-making process is crucial. The agents learn how to transition from the initial state to one that is more conducive to achieving the goal. The goal is to locate From the start of the movement, the shortest and optimal route to the destination.


FIGURE 2.5: Reinforcement Learning process visualisation

### 2.4.1 Problem formulation in RL

A RL problem begins with an agent that interacts with the environment and defining its state $s$, when the agent initiates an action $a$ that leads him to the next state $s+1$, it also gets a variable quantity that represents the reward $r$. The reward or prize is letting know the agent of how well executed the action is at that instant by also taking in consideration the performance measure, the aim of the agent is to slowly learn the strategy that maximizes the reward as a result of the actions taken. The agent's action making is what will we call a policy $\pi$ . A value function $v$ is a function of state that estimates how good it is for the agent to be in a state, or how good it is for the agent to perform an action while being in a particular state.

- The State-Value function for a policy $\pi$, denoted as $v_\pi$ tells us how well it is for the agent to be in a state given a certain policy $\pi$.
- The Action-Value function for a policy $\pi$, denoted as $q_\pi$ gives us a value under a certain policy that tells us how well it is for the agent to take

action while being in a certain state, the $q_\pi$ is also referred as the Q-function in general [16].

### 2.4.2 Shortcomings of RL

RL does have shortcomings, in fact when the number of states computed is becoming too large then that leads to two shortcomings [17]. First of all, the computational cost is greater to explore all the state and action possibilities to identify the optimal actions and thus having a slow learning time. Secondly, a large storage capacity is then required to store all this knowledge.

### 2.4.3 Q-Learning

One of the simplest models and arguably one of the most applied representative in RL is a model-free single-agent RL approach called Q-learning (see figure 2.6), its goal is to find the optimal policy by learning the optimal Q-values for each state-action pair. The Q-table is the main component of Q-learning , the table is a matrix where each element is identified as a Q-value which represents the value of each state-action pair. The algorithm evaluates Q-value of the Q-table by the agent of Q-learning then will receive a reward or penalty for every action a taken in state s. In every iteration steps, the maximum Q-value at state s will be selected by the Q-learning agent. Then, evaluation from the reward function for that action will be stored in the Q-table when the algorithm moves to the next state [16].

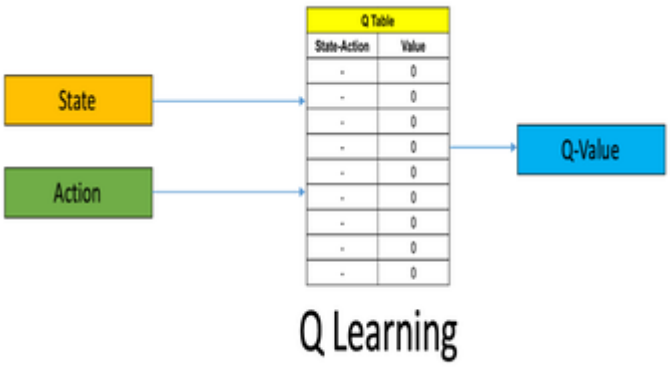


FIGURE 2.6: Q-learning visualisation

## 2.5 Deep Reinforcement Learning

Deep reinforcement learning refers to reinforcement learning with neural networks as a function approximators and that's where combining DL techniques with RL can provide the solution needed to tackle the shortcomings,

DRL grasps the concepts of deep neural networks with it enabling a continuous state-space representation allowing large number of states. DRL also reduces the learning time required to explore all state-action pairs and identify optimal actions. Finally, DRL uses several layers of neurons to store the weights of the links connecting the neurons, which are used to approximate the Q-values efficiently in order to address the storage issue in RL [18].

### 2.5.1 Deep Q-learning

An adaptation of the Q-learning approach previously seen with deep learning techniques is called Deep Q-Learning it uses a deep neural network to approximate the values (see figure 2.7). The initial state the agent perceives is absorbed by the neural network as input and it proceeds by returning the Q-value for all possible action-state as an output. it also stores all previous experience in memory and performs training on the network with the gained experience [2].

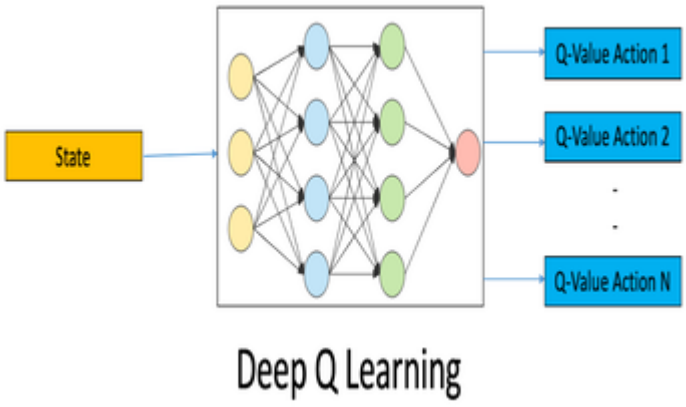


FIGURE 2.7: Deep Q-learning visualisation

## 2.6 Conclusion

In this chapter, a brief overview of how convolutional neural networks have achieved great advancements in different field of applications. A review of some basic understanding about reinforcement learning was done, presenting Q-learning combining it with deep learning.

# Chapter 3

# An enhanced deep reinforcement learning algorithm

## 3.1 Introduction

The regulation of traffic lights at road intersections is the focus of this thesis. In this chapter, we will first outline the work of [1] as it represents the basis of our model. Then we will detail the improvements made to this solution in order to improve its performances.

## 3.2 Q-learning algorithm to adjust traffic signal timing [1]

The problem is formulated as a reinforcement learning (RL) problem in which an agent interacts with the junction at discrete time steps $t = 0, 1, 2, ..$etc, with the agent's long-term objective being to decrease the amount of time vehicles spend at the intersection.
To be more specific, the RL agent observes the intersection and determines the state $S_t$ and then the agent moves to the next state $S_{t+1}$ once a car crosses under actuated traffic after a certain action $A_t$.
As a result of the agent's choice of traffic lights, it also receives reward $R_t$ at the conclusion of time step $t$ hence the sequence ($S_t$,$A_t$,$R_t$,$S_{t+1}$), such a reward acts as a cue directing the agent to accomplish its objective, the reward is a single integer with a variable value serves as the prize.
The agent's informal objective is to maximize the overall reward it gets, this entails optimizing long-term cumulative reward in addition to immediate reward.

### 3.2.1 Problem environment

The authors in [1] modeled the road as a four way intersection using Sumo [19] that is a multi-modal traffic simulation software that is open source, extremely portable, tiny, and continuous that can manage large networks, where each way or direction has 4 lanes; 3 lanes leading to the intersection (Figure 3.1).

Each lane taking the vehicles down its direction ; right lane to go right, middle lane to stay at the middle and the left lane to go left at the intersection.
The green and yellow light period are 10 seconds and 6 seconds respectively.

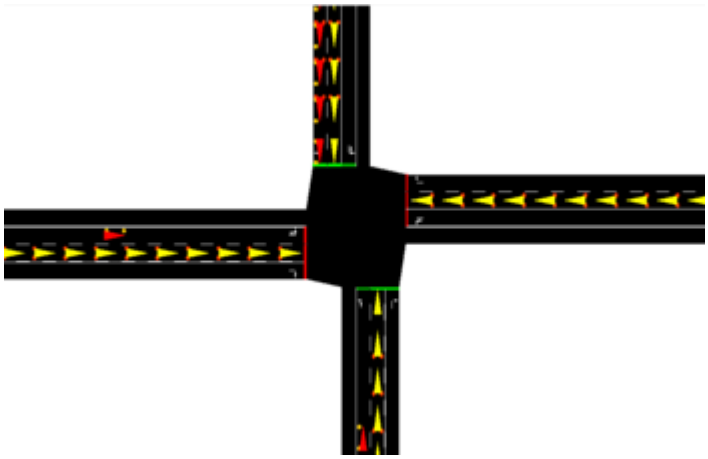

FIGURE 3.1: Intersection Model

### 3.2.2 Defining State, Action and Reward

#### State definition

The approach the authors [1] used to define the state is for each road way is to have to generate velocity $V$ and position $P$ matrices (see Figure 3.2).

#### Action set definition

After the RL agent sees the already defined state, it has to choose actions to move to the next state.
The set of actions decided was two;

- Turning on green lights for the horizontal roads
- Turning on green lights for the vertical roads

Between two states, the agent goes through a transition that involves :

- Change the lights for vehicles going straight to yellow
- Change the lights for vehicles going straight to red
- Change the lights for vehicles turning left to yellow
- Change the lights for vehicles turning left to red

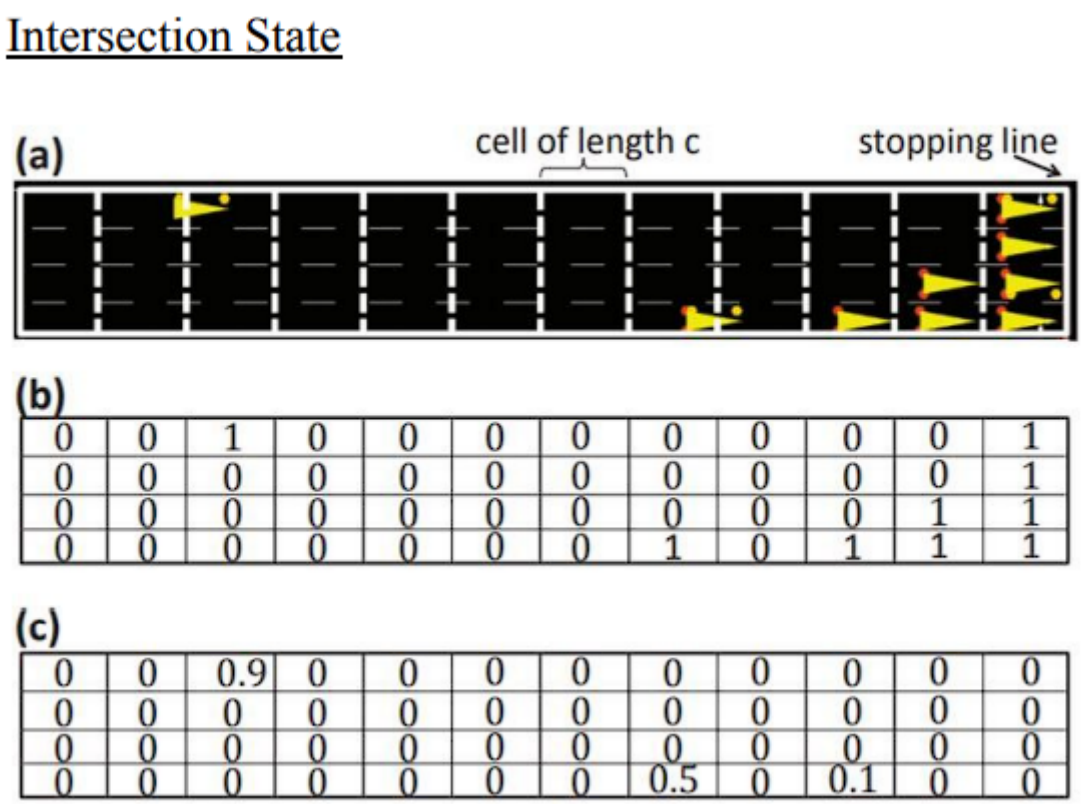


FIGURE 3.2: Intersection State

**Reward definition**

The reward is given to the agent at every time step, it takes two observations of the waiting time of all vehicles once at the beginning of the time step $R_1$ and another at the end $R_2$.
The total reward $R_T$ will be the first observation minus the second observation, if the outcome is positive by decreasing the waiting time it rewards the agent else if it's negative then it's punishing the agent because the waiting time increases.

### 3.2.3 Q-learning algorithm proposed by [1]

The authors in [1] had to work with an unlimited number of states, hence it is impossible to save the Q-value for every single state. As a result, they used an approximation method called the Bellman Optimality Equation to calculate the Q-value for the given state (see Figure 3.3).

$$Q^*(s,a) = \mathbb{E}\Big\{ R_t + \gamma \max_{a'} Q^*(S_{t+1}, a') | S_t = s, A_t = a \Big\}$$
$$\text{for all } s \in \mathcal{S}, a \in \mathcal{A}$$

FIGURE 3.3: Bellman Optimality Equation

To approximate the Bellman equation, they trained a deep neural network as shown in Figure 3.4:

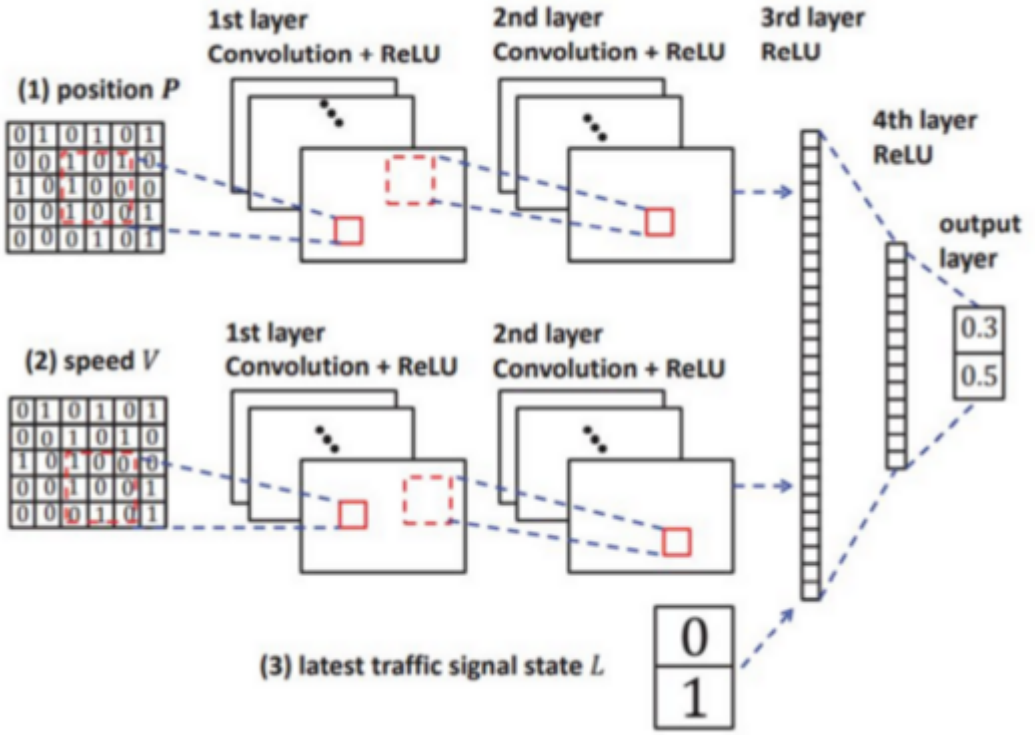


FIGURE 3.4: Deep Neural Network Structure

The deep neural network introduced has three inputs matrices, one for the position $P$, another for the speed $V$ and finally the state of where the network is at $L$.
The position $P$ and the speed $V$ matrices undergo two layers of convolution accompanied by the activation function known as ReLU, then passing by the third and fourth layer with both their activation function as ReLU.
Finally, the last layer which is a linear layer outputs the Q-values according to each potential course of action the agent takes.

**Training the neural network**

The neural network was trained by first initializing random weights to it, when every time step begins, the agent observes the current time step $S_t$,gathers data for the neural network, and executes the action $A_t$ with the highest potential future reward and then extracting training examples from the couples state and action ($S_t$,$A_t$) in order for the agent to learn by minimizing a following loss function which its equation is as follows :

$$MSE(\theta) = \frac{1}{m}\sum_{t=1}^{m}\Big\{\big(R_t + \gamma \max_{a'} Q(S_{t+1}, a'; \theta')\big) - Q(S_t, A_t; \theta)\Big\}^2$$

FIGURE 3.5: Mean Squared Loss Function

where $m$ denotes the size of the input, the authors chose to use stochastic gradient descent algorithm RMSProp with a mini-batch of size of 32.

## 3.3 An enhanced deep learning algorithm : our contribution

In this section we will address the improvements done to the algorithm of [1] in order to improve its performances.

### 3.3.1 Learning rate and Batch size

One of the first major changes we did on the implemented algorithm by [1] was decreasing step by step the batch size of the mini-batch size type used while also decreasing the learning rate of the algorithm to find the optimal values for these parameters that help decrease the wait time of vehicles and thus improving the overall performance of the algorithm.

In fact [20] concluded that there is a high correlation between the learning rate and the batch size values, and recommended to use small learning rates with small batch sizes.
We decreased the learning rate from *LR*=0.0002 to *LR*=0.0001 as well as the batch size which was *batch size*=32 down to a size of *batch size*=2.

### Learning Rate

The concept of learning rate is used a lot in machine learning it refers in essence, to the speed with which an algorithm finds a solution (see Figure 3.6).
It's one of the most crucial hyper-parameters for training neural networks. Therefore, it's crucial to configure its value as closely as feasible to the ideal. If the learning rate chosen is too small or too low then the gradient descent will take a huge amount of time to reach the optima on the other hand, if the learning rate is too high, the gradient descent may begin to diverge and fail to find the best solution.

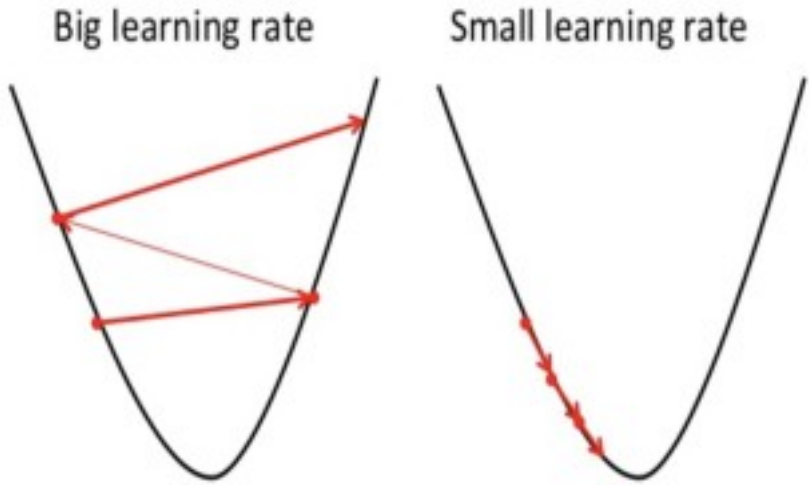


FIGURE 3.6: Significance of the learning rate value

### Batch Size

What the batch size is doing is defining how many samples it is using to train the neural network in a single epoch, an epoch is when a neural network processes a whole dataset just once, both forward and backward. There are many different types when coming to the gradient descent as shown in Figure 3.7:

- Batch Gradient Descent : This type of batch requires significant amount of memory if our training set contains huge amount of samples and can only update the model a few times before reaching the local minima. However there is a big drawback to using batch, we could become trapped in a saddle point too early and end the training with parameters that are far from the required performance for non-convex problems with several local minima.

- Mini-Batch Gradient Descent: The most commonly used and the one showing the best result is the mini-batch which the implemented algorithm already uses. Using the mini-batch, the model is updated more

frequently compared to using only batch. it also solves the issue for problems with several local minima and allows for a strong convergence.


FIGURE 3.7: Types of gradient descent

### 3.3.2 Epoch

Another one of the improvements we did was increasing the epoch parameter, as seen earlier an epoch is an a hyper-parameter that specifies how many samples must be processed before the internal model parameters are updated.
The implemented algorithm used a number of *epoch*=1 and that means, only one instance of the dataset is run forward and backward through the neural network.
We found it too little it can furthermore cause an under fitting scenario, under fitting happens when the model constructed is too simple that can neither model the training data nor generalize to new data (see Figure 3.8).

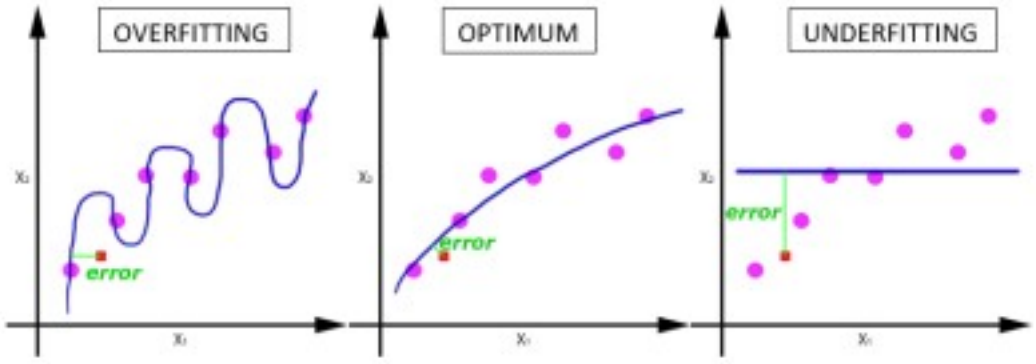


FIGURE 3.8: The impact of different Epoch values on the model

To make the model more accurate and generate less errors we increased the number of Epoch value, as the number of the Epoch value increases the neural network's weights are adjusted more often, and the curve shifts from

under fitting to optimum until we converged to an optimum scenario and its value *Epoch*=10.

### 3.3.3 Activation Function

In artificial neural networks, activation functions are utilized specifically to convert input signals into output signals that are then sent as input to the following layer in the model (see Figure 3.9).
In the proposed algorithm by Tej Patel and Robert Cook in [1], The ReLU activation was applied for the first two layers of the convolutional neural network. After experimenting with different activation functions we proposed to use an improvised version of the ReLU function called LeakyReLU.
The authors of [21] actually provide a helpful explanation of the many activation functions that are employed in the field of deep learning, as well as their significance in creating a deep learning model that is both successful and efficient, we will present some notable functions as well as explain why the choice of using LeakyReLU was made.

#### Linear Activation Function

The output of the linear activation function is directly related to the input, its equation can be defined as follows :

$$f(x) = ax$$

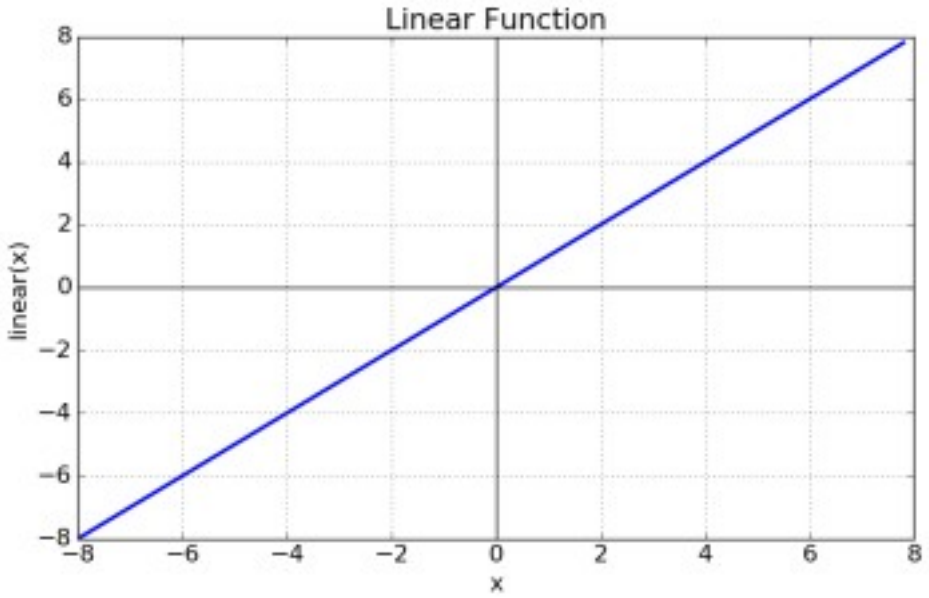


FIGURE 3.9: Linear Activation Function

Using the linear activation function won't benefit the neural network because simply put the function won't improve the error, in the other hand it's used in the final layer when all the layers collapse into one.

### ReLU Activation Function

The rectified linear unit or as so called ReLU is a non linear function that does not require any intensive processing because there is no complex math involved (see Figure 3.10). as a result, the model can be trained faster and learn faster too.
The ReLU activation function's equation is as follows:

$$f(x) = max(0, x)$$

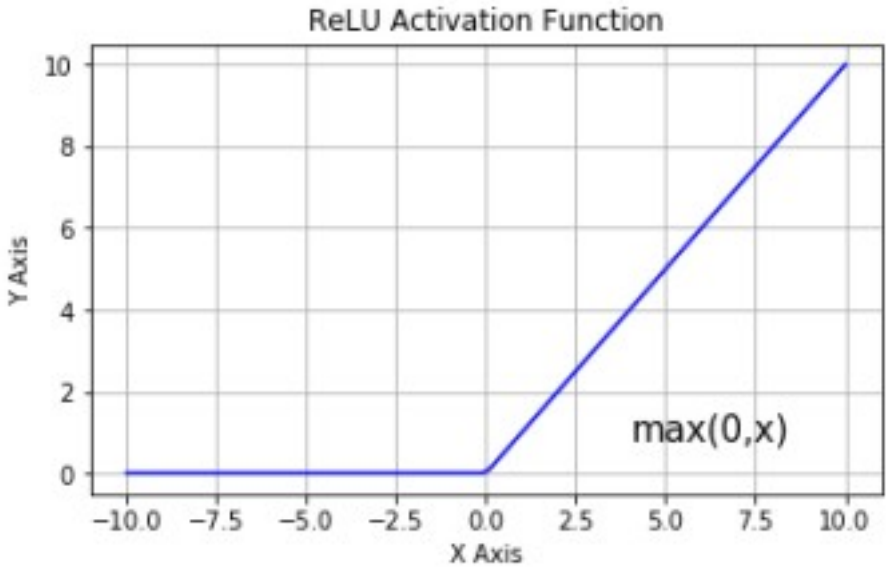


FIGURE 3.10: The ReLU Activation Function

ReLU function does have limitations, the most important one is the issue of the dying ReLU, that happens when a ReLU neuron is caught on the negative side and consistently produces 0. It is unlikely that a neuron would recover after it has gone negative because the slope of ReLU in the negative region is likewise 0. These neurons have no role in processing the information, hence they are effectively useless.
What can happen sometimes when the learning rate is too high, is that a good chunk of neural network practically dies. So we used the LeakyReLU function that solves this problem in case it happens.

### LeakyReLU activation function

The LeakyReLU activation function is a variation of the ReLU function that solves the problem of the dying ReLU (see Figure 3.11).
Instead of the ReLU function's value being 0 for negative x values, it is specified as a very small linear component of x, its mathematical equation is :

$$f(x) = ax, x < 0$$

$$f(x) = x, x >= 0$$

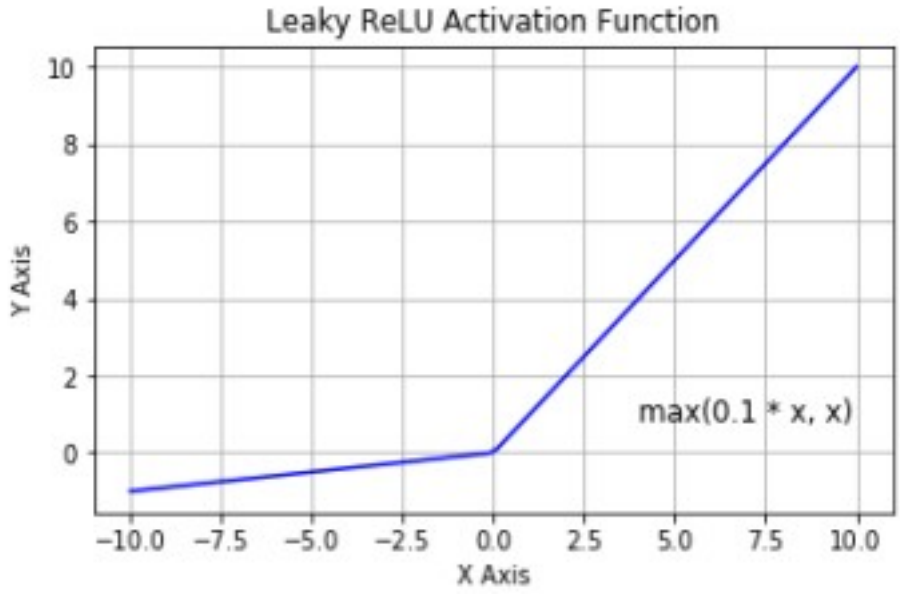


FIGURE 3.11: The LeakyReLU Activation Function

### 3.3.4 Adam Optimizer

As seen earlier, the RMSProp optimization technique was used in the already implemented algorithm for training the neural network.
It is very well-known despite not having been published in a formal academic paper by its developer Geoffrey Hinton, RMSProp is used coupled with gradient descent to update the latter step by step.
RMSProp works by dividing the learning rate by the average of the squared gradients with exponential decay which means that it doesn't decay its' learning rate too quickly like some other techniques before the introduction of RMSProp.
Another more interesting optimizer introduced by [22] called the Adam optimizer was added in our improvements instead of RMSProp, the method improves the RMSProp optimizer by adding the notion of momentum to it.
The concept of momentum is used to accelerate gradient descent by using the exponentially weighted average of the gradients, and that helps converging into a minima way more quickly.
The Adamax method is just another variant also proposed by the authors, but achieves lesser accuracy than Adam.

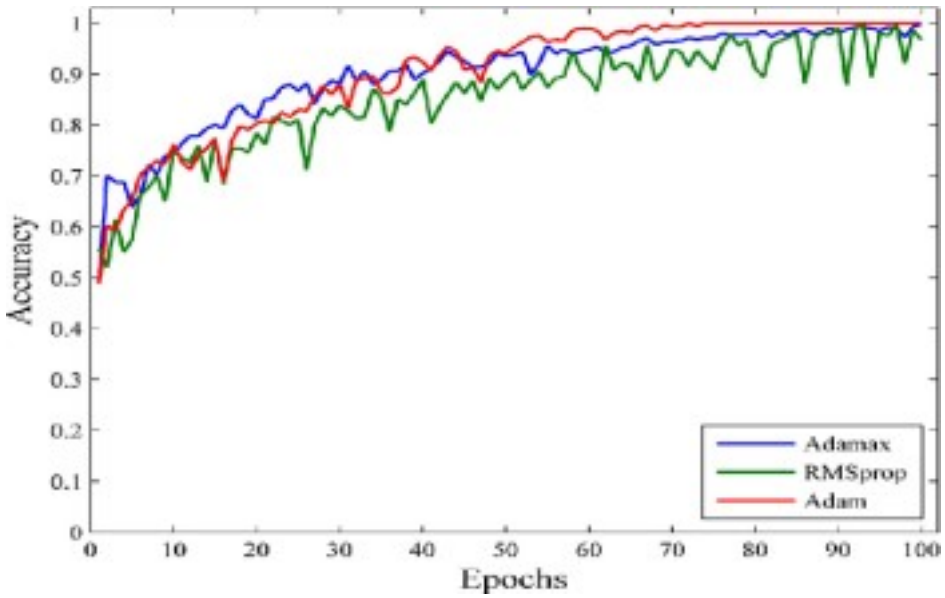


FIGURE 3.12: A brief accuracy comparison between RMSProp, Adam and Adamax

### 3.3.5 Batch Normalization

Batch normalization is considered a layer that uses a normalization technique between the layers of neural network, the work [23] investigated how batch normalization improves the overall accuracy of the neural network as well as the speed rate at which it is trained (see Figure 3.13).
In addition, they have shown that the outputs of unnormalized networks are huge and can sometimes poorly behave, it is in this context that we implemented a batch normalization layer after every activation function in our work.

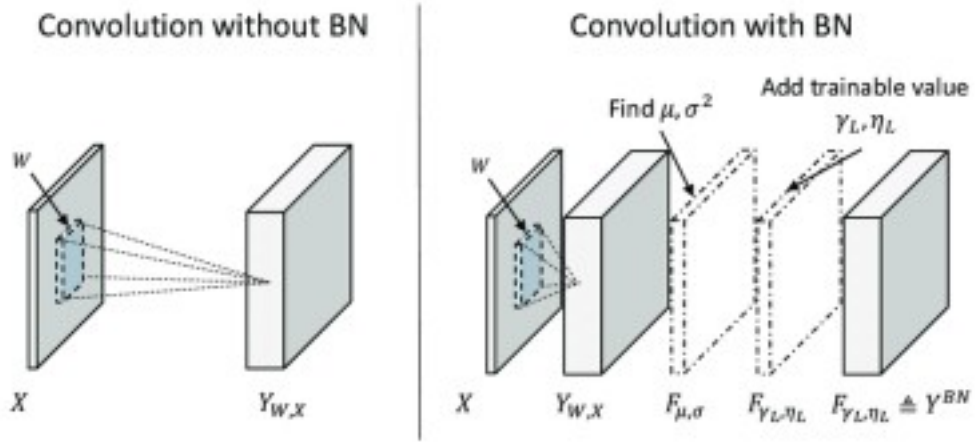


FIGURE 3.13: Description of Batch Normalization

### 3.3.6 Dropout Layer

We have seen earlier that one of the goals is increasing the algorithm's performance by preventing the overfitting problem, one of the other layers that helps solves that issue that we added beside the batch normalization layer is the dropout layer. A dropout layer as explained by [24] is a layer that applies an algorithm, simply put the technique nullifies certain neurons' contribution to the subsequent layer while maintaining the integrity of all other neurons.
Following the 2012 paper [25], it was recommended to use dropout with a rate of $p$=0.5 for each fully connected layer before the output, in our case after the third and the fourth layers. The authors of [26] even recommended to use dropout after every convolutional neural network layer with a lower rate (see Figure 3.14).

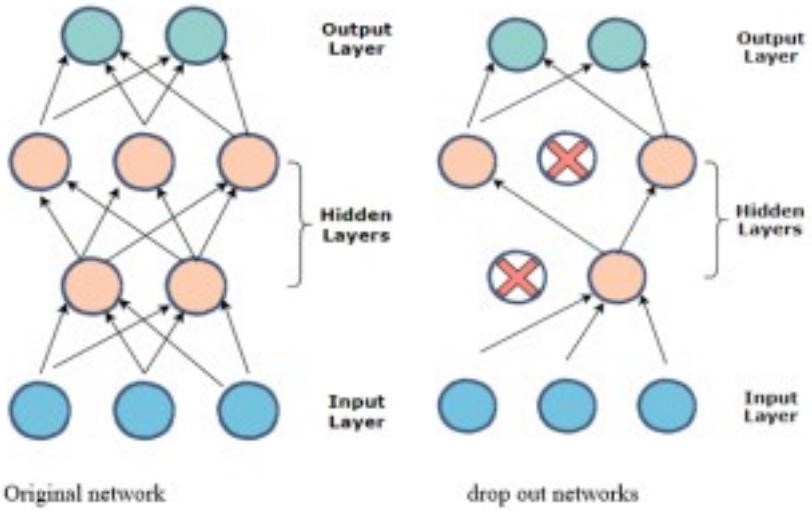


FIGURE 3.14: Comparison over a neural network with dropout and one without dropout

### 3.3.7 Conclusion

In this chapter, we presented the approach to how the problem was formulated in order to have a better basis understanding of the operating principle, then quickly moved on to the structure of the algorithm adopted by the algorithm's developers.
Then we proposed many improvements to the algorithm, by adding some layers (Batch normalization, Dropout) to the convolutional neural network and changing the values of some important hyper parameters (Learning rate, mini-batch size, Epoch, Activation function, Adam Optimizer), assessing the difference in performance between the algorithm and our improved approach is the topic of the next chapter.

# Chapter 4

# Evaluation of performance

## 4.1 Introduction

We provided a thorough description of our proposal in the preceding chapter. This chapter will outline the proposed improvements by demonstrating that our solution, has an overall better performance in the average waiting time compared to the already implemented solution. But firstly, we will provide a brief description of the many tools used to simulate the environment as well as the various libraries employed.

## 4.2 Tools and Resources

In what follows, we will outline the most important tools used.

### 4.2.1 Simulation Tool : SUMO

To simulate the environment, an open source, portable, microscopic, and continuous multi-modal traffic simulation tool called Simulation of Urban MObility (SUMO) is made to simulate and manage massive traffic networks (see figure 4.1).

FIGURE 4.1: SUMO environment

#### NetEdit

Editing the environment we worked on, can only be done using netedit. Which is a portable graphical network editor included in SUMO.
It can be used to create networks from scratch and to modify all aspects of existing networks. With a powerful selection and highlighting interface it can also be used to debug network attributes (see figure 4.2).

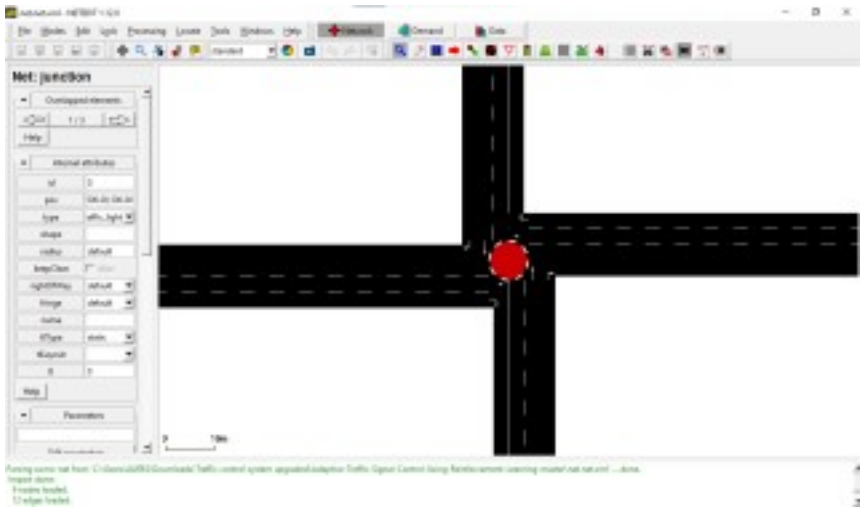

FIGURE 4.2: NetEdit environment

### 4.2.2 Developing Tools

To make the necessary improvements for our solution, we used many developing modules and packages offered in Pycharm's integrated developement environment.

#### Pycharm

In the case of our model, python programming was only done in Pycharm, Pycharm is an integrated development environment. It has a graphical debugging tool and allows code analysis. Additionally, it supports web development using Django and allows for the management of unit tests as well as the integration of version control software.

#### TraCI

TraCI uses a TCP/IP architecture to retrieve some crucial information after every simulation. TraCI is the short term for "Traffic Control Interface". Giving access to a running road traffic simulation, it also allows us to manipulate their behavior "on-line".

#### Tensorflow

The Google Brain team first created TensorFlow. In the beginning, the goal was to use neural networks to enhance Google services like Gmail, Photos, and the search engine. With the help of this framework, researchers and developers could collaborate on IA models. TensorFlow was then made public for the first time at the end of 2015.

#### Keras

Keras is an API, it enables interaction with algorithms. We use it to create models, access optimizers, Optimizers are classes or methods used to change the attributes of your machine/deep learning model to help to get better results faster.

#### Sumolib

For interacting with sumo networks, simulation output, and other simulation artifacts, there is a set of python modules called sumolib.

#### Numpy

One of the most fondamental Python module used for scientific computing is called NumPy. A multidimensional array object, various derived object, and a variety of routines for quick operations on arrays are provided by this Python library.
These operations include discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation, and much more.

## 4.3 Simulation Settings and Results

In this section, We will verify our algorithm's simulation settings. Then firstly, compare it with our constructed implementation of a static traffic light control system in terms of waiting time. Secondly, a comparison will be done between our improved solution and the original solution.

### 4.3.1 Simulation Settings

We employ Simulation of Urban MObility (SUMO), a well-liked open source simulator. We used for all the scenarios ; static traffic light control, implemented algorithm by [1] and our improved proposition.

#### Intersection

We consider an intersection of four ways, each way having four lanes. Three lanes for leading into the intersection, and a last one for carrying out the traffic out of the junction.

#### Traffic

East-west traffic is more likely to be generated by the traffic arrival process than north-south traffic, which is probabilistic.
Over the course of the simulation roughly 1000 to 1200 vehicles will cross the intersection. The green light and yellow light duration do not change, 10 seconds for green light and 6 seconds for the yellow light.
The maximum speed allowed for vehicles for all the lanes is around 13.89 *m/s*, around 50 *k/h*

#### Simulation Data Processing

For all 77 episodes we did of our simulations for each algorithm, we recorded the waiting time for all vehicles. We then estimated an average waiting time for each scenarios, and then compared them side by side. In case of the original work, the simulation of one episode duration was around 40 minutes. Ours was around 5-7 minutes for one episode.

### 4.3.2 Simulation Results

First, we saw that the original work [1] showed that when using a static light configuration, They get a waiting time of 338798 seconds. Even though there was no said implementation coming from them when viewing the work.
As a result, we tried to replicate the same result by implementing a static configuration, but we quickly found that the waiting time for the vehicles at the intersection was even much higher 1331436 seconds than expected (see figure 4.3).

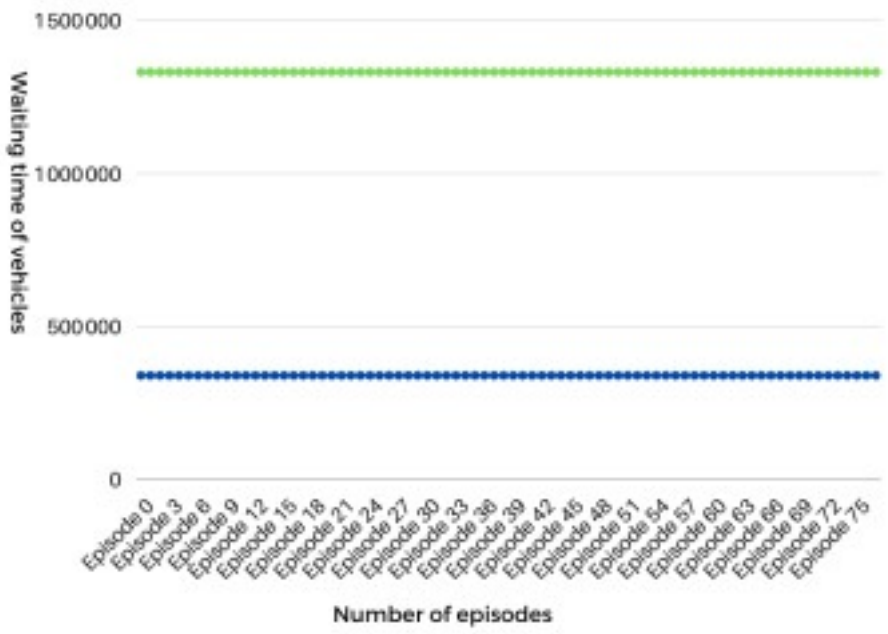


FIGURE 4.3: Comparison between our implemented results(light green) and the results shown(blue) for a static configuration

Next, We compared the waiting time of our enhanced deep reinforcement learning algorithm to the original one. We found out that our algorithm has a more stable behaviour, and the value of waiting time can even go down under the 100000 seconds threshold. In fact we could see our model starts up with the highest waiting time but it decreases quickly. Even at some episodes(12,19,30,37,67), we found out that the waiting time spikes up but it always rapidly comes back to normal (see figure 4.4) and see figure 4.5).
By comparison, the original work did worse. In actuality, the algorithm did start with a little lower value than ours, but it didn't maintain a decrease in the waiting time.
Unfortunately, after the second episode (episode-1) the value increased until episode-11 where it went down to 170000 seconds. Even then, the algorithm could not maintain a firm reduction in the waiting time, It did hit randomly under 100000 seconds but the values kept fluctuating extremely.

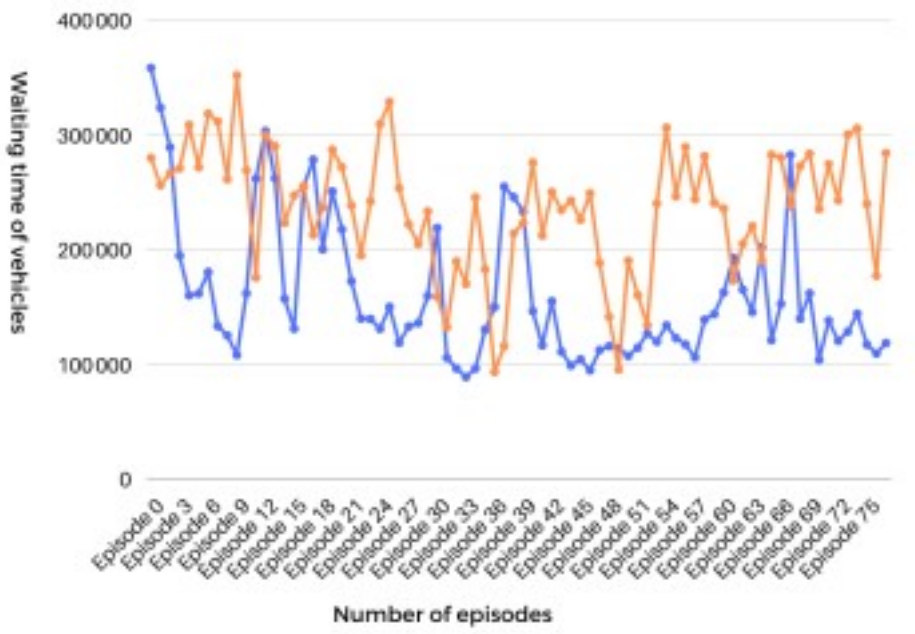


FIGURE 4.4: Comparison of the waiting time of our enhanced approach (blue) and the already existing algorithm (orange)

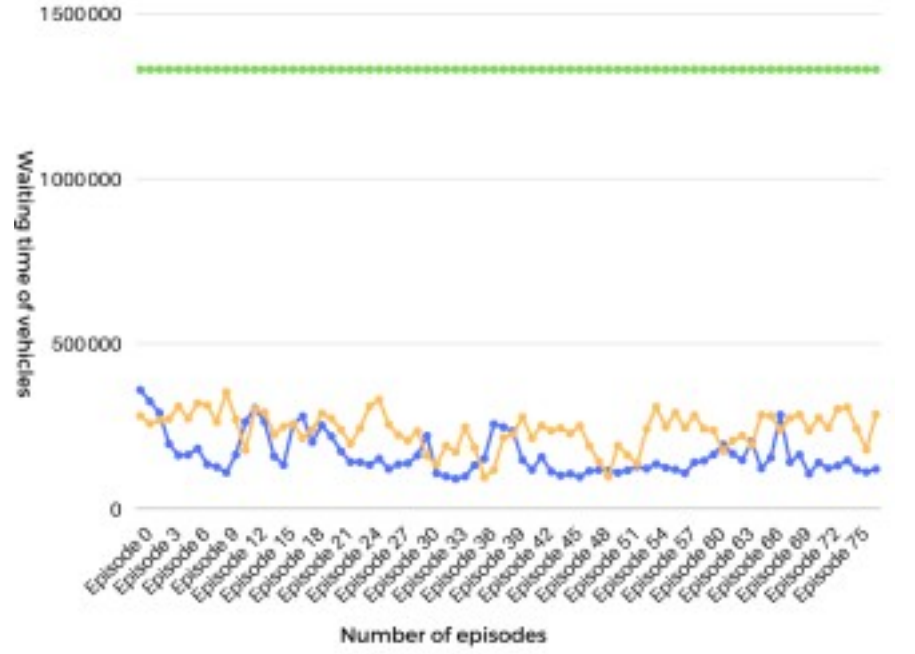


FIGURE 4.5: Comparison of the waiting time of our enhanced approach (black), the already existing algorithm (light blue) and the static light configuration (orange)

**Average waiting time metric**

Finally, for the 77 episodes we calculated the average waiting time for each algorithm. The average waiting time for the original deep reinforcement learning algorithm was about 237026 seconds, our enhanced deep reinforcement learning algorithm proved a noticeable increase in performance. In fact the average waiting time for our proposed solution was 161102 seconds.
By taking the average waiting time values for each algorithm into account, we can say that our algorithm had a 32.03% increase in performance compared with the original implemented work.

## 4.4 Conclusion

In this chapter, we defined the libraries and tools used for simulating and improving the original work.
We then showed all the simulation results, from the result of each episode giving us the waiting, to the average waiting time of both models. Proving that our enhanced deep algorithm out performs the existing algorithm by a substantial margin.

# General Conclusion

It is well known that traffic congestion has resulted in some major socioeconomic issues. In the last decades, machine learning techniques have shown promising results when used effectively, that's why a lot of researches had been done in the field of traffic light control systems.

Within this context, we proposed in this thesis an enhanced deep reinforcement learning algorithm based of an already implemented work for traffic light control systems. Our aim was to reduce more effectively and in a more stable way, the overall waiting times of vehicles at any given intersection.

To achieve this objective, we have proposed to adjust some key parameters, namely learning rate, mini-batch size, Epoch, Activation function, Adam Optimizer as well as adding some important layers to the convolutional neural network.

We conducted with success several simulations by varying the values of different parameters. The obtained results showed that our algorithm significantly reduces the overall waiting time of vehicles and performs better roughly by 30% with regards to the original algorithm. Furthermore, we found out that deep reinforcement learning can have some stability issues.

Due to time constraints, we were only able to use one intersection during the simulation. As perspective, we propose to adapt our algorithm to take into account several crossings which are of relevance. And why not try to implement it in the real world as a perspective in order to benefit the public.

# Bibliography


[1] Tej Patel et al. "A Q-Learning Approach to Traffic Light Signal Control". In: *Github* (2018). URL: https://github.com/TJ1812/Adaptive-Traffic-Signal-Control-Using-Reinforcement-Learning.

[2] Xiaoyuan Liang et al. "A Deep Q Learning Network for Traffic Lights' Cycle Control in Vehicular Networks". In: *IEEE Signal Processing Magazine* (2018).

[3] Ng S. C. Kwok C. P. "An intelligent traffic light system using object detection and evolutionary algorithm for alleviating traffic congestion". In: *International Journal of Mathematical and Computational Sciences* (2020).

[4] Juntao Gao Yulong Shen et al. "Adaptive Traffic Signal Control: Deep Reinforcement Learning Algorithm with Experience Replay and Target Network". In: (2017).

[5] A. J. Miller. "Settings for fixed-cycle traffic signals." In: *Settings for fixed-cycle traffic signals.* 14 (1963), pp. 373–386. URL: https://doi.org/10.2307/3006800.

[6] Z. R. Abdy. "Fuzzy logic traffic signal control". In: ().

[7] Xiaoyan Yang Shuguo Yang. "The application of the queuing theory in the traffic flow of intersection". In: *International Journal of Mathematical and Computational Sciences* 8.6 (2014).

[8] Roger P.Roess et al. "Traffic Engineering : Third Edition". In: *Pearson Education International* (2004).

[9] Nathan David Chinedu Duru. "Evaluating the Emission of CO2 at Traffic Intersections with the Purpose of Reducing Emission Rate Case Study: The University of Nigeria Nsukka Evaluating the Emission of CO2 at Traffic Intersections with the Purpose of Reducing Emission Rate, Case Study: The University of Nigeria Nsukka". In: *International Journal of Environmental Science and Sustainable Development* (2018). URL: 10.21625/essd.v3iss2.373.

[10] Ivan Nunes da Silva Danilo Hernane et al. "Artificial Neural Networks : a practical course." In: (2017).

[11] Sakshi Indoliaa Anil Kumar Goswamib S. P. Mishrab Pooja Asopaa. "Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach". In: *Procedia Computer Science* (2018).

[12] Y. LeCun et al. "Handwritten digit recognition with a back-propagation network". In: *Advances in neural information processing systems* (1990).

[13] A. Krizhevsky et al. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* (2012).

[14] C. Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015).

[15] A. G. Barto et al. "Introduction to reinforcement learning." In: *MIT press Cambridge* 135 (1998).

[16] Beakcheol Jang Myeonghwi Kim et al. "Q-Learning Algorithms: A Comprehensive Classification and Applications". In: *IEEE Xplore* (Sept. 2019). URL: 10.1109/ACCESS.2019.2941229.

[17] H. L. Khoo et al. "A survey on reinforcement learning models and algorithms for traffic signal control". In: *IEEE Xplore* 50.34 (Oct. 2017), pp. 1–38. URL: https://doi.org/10.1145/3068287.

[18] Marc Peter et al. "A Brief Survey of Deep Reinforcement Learning". In: *IEEE Signal Processing Magazine* (Sept. 2017), p. 16. URL: https://arxiv.org/pdf/1708.05866.pdf.

[19] "Eclipse SUMO". In: *Github* (2001). URL: https://github.com/eclipse/sumo.

[20] Ibrahem KandelMauro Castelli. "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset". In: *ScienceDirect* (May 2020). URL: https://www.sciencedirect.com/science/article/pii/S2405959519303455.

[21] Siddharth Sharma et al. "Activation functions for neural networks". In: *International Journal of Engineering Applied Sciences and Technology* 4.12 (Apr. 2020), pp. 310–316. URL: https://www.sciencedirect.com/science/article/pii/S2405959519303455.

[22] Diederik P. Kingma Jimmy Lei Ba. "Adam : A method for stochastic optimization". In: *Seoul National University* (2015). URL: https://arxiv.org/pdf/1412.6980.pdf.

[23] Johan Bjorck et al. "Understanding of Batch Normalization". In: *NeurIPS Proceedings* (2018).

[24] Pierre Baldi Peter Sadowski. "Understanding Dropout". In: *Seoul National University* (2016).

[25] Pierre Baldi Peter Sadowski. "Improving neural networks by preventing co-adaptation of feature detectors". In: *Seoul National University* (2016).

[26] Sungheon Park and Nojun Kwak. "Analysis on the Dropout Effect in Convolutional Neural Networks". In: *Seoul National University* (2016).