The People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
AMAR TELIDJI LAGHOUAT UNIVERSITY

FACULTY OF SCIENCE
COMPUTER SCIENCES DEPARTMENT

Master Thesis

DOMAINE : Mathematics and COMPUTER SCIENCES

FILIERE : COMPUTER SCIENCES

Option : Network Engineering and Distributed System

By :

Idrissi Moulay Ali **Labiodh Hachani**

Theme

Vehicle Trajectory Prediction in Urban Traffic Networks using Deep Learning

Proposed by:-Mme Hadda Cherroun - Mr Bouzaine Brik

Presented in front of the jury composed of :

- Mr Guellouma Younes MCA (University Amar Telidji Laghouat) President

-Mr Nahari Attia MCB (University Ziane Achour Djelfa) Examinator

-Mme Hadda Cherroun Professeur (University Amar Telidji Laghouat) Supervisor

Academic year 2020/2021


بسم الله الرحمن الرحيم

DEDICATIONS

With God's help, I was able to carry out this modest work that I dedicate

To my dear parents, with all my respect and love, as a sign of affection, gratitude and gratitude for all their encouragement and sacrifice throughout my studies. I hope that this work will be the fruit of their love. I hope that you are proud of me and that God will keep you.

To my sisters Khadidja, Leila, Iman, Aicha and also my friends Abdou, Mohamed, Bachir, Moulay, you are the light that illuminates the dark days in my life.

To all those I hold dear.

Labiodh Hachani

DEDICATIONS

With God's help, I was able to carry out this modest work that I dedicate

To my dear parents, with all my respect and love, as a sign of affection, gratitude and gratitude for all their encouragement and sacrifice throughout my studies. I hope that this work will be the fruit of their love. I hope that you are proud of me and that God will keep you.

To my sisters Anfel, Amina and also my friends Aymen, Ahmed, Moussa, Reda, Hosine, Bachir, Hachani, Hamid, Ilyes, Ali, Ayoub you are the light that illuminates the dark days in my life.

To everyone I care about.

Idrissi Moulay Ali

THANKING

In the first place, we thank Allah for giving us the will, health, strength and courage to complete this work.

We would like to thank all the teaching team who followed us and trained us during our course and especially our professors . Hadda Cherroun and Mr. Bouzaine Brik , we would also like to express our gratitude for his patience and support, which have been invaluable to us in carrying out our work.

I would like to extend my heartfelt thanks to Department Head Mr. BENSAAD Lahcen, and the jury members for agreeing to judge this modest work.

Finally, our thanks go to everyone who has contributed directly or indirectly to this work.

ABSTRACT

Our work focuses on vehicles positions prediction which is fundemental problem that can be deployed in many important smart vehicles technologiesand services related.
The main objectives are two folds. First we investigate Deep Learning (DL) based models to generate prediction .Second objectif concerned wether we can relay on artificial (simulated) data to generate accurate models we needed.
To do that we have designed two Deep Learning (DL) approches LSTM and CNN. In the experiments we have used a set of tools; keras and several python librairies. concerning the data we have used San Francisco mesured data and Sumo simulator to generate artificial data.
The results showed that LSTM model is a little more efficient than CNN, also result showed that artificial data has a suitable degree of reliability.

RÉSUMÉ

Notre travail se concentre sur la prédiction des positions des véhicules qui est un problème fondamental qui peut être déployé dans de nombreuses technologies et services liés aux véhicules intelligents importants.

Les objectifs principaux sont doubles.Tout d'abord, nous étudions les modèles basés sur le Deep Learning (DL) pour générer des prédictions. Le deuxième objectif concernait si nous pouvons nous appuyer sur des données artificielles (simulées) pour générer des modèles précis dont nous avions besoin.

Pour ce faire, nous avons conçu deux approches de Deep Learning (DL) LSTM et CNN. Dans les expériences, nous avons utilisé un ensemble d'outils ; keras et plusieurs librairies python. concernant les données, nous avons utilisé les données mesurées de San Francisco et le simulateur Sumo pour générer des données artificielles.

Les résultats ont montré que le modèle LSTM est un peu plus efficace que CNN, les résultats montrent également que les données artificielles ont un degré de fiabilité approprié.

ملخص

يركز عملنا على التنبؤ بأوضاع السيارة وهي مشكلة أساسية يمكن نشرها في العديد من تقنيات وخدمات المركبات الذكية الهامة.

الأهداف الرئيسية ذات شقين: أولاً، نقوم بدراسة النماذج القائمة على التعلم العميق (DL) لتوليد التنبؤات. كان الهدف الثاني هو ما إذا كان بإمكاننا الاعتماد على بيانات اصطناعية (محاكاة) لإنشاء نماذج دقيقة نحتاجها.

للقيام بذلك، قمنا بتصميم طريقتين للتعلم العميق (DL) LSTM وCNN. في التجارب استخدمنا مجموعة من الأدوات ؛ keras والعديد من مكتبات بيثون.فيما يتعلق بالبيانات استخدمنا البيانات المقاسة من سان فرانسيسكو ومحاكي SUMO لإنشاء بيانات اصطناعية.

أظهرت النتائج أن نموذج LSTM أكثر كفاءة بقليل من CNN ، وأظهرت النتائج أيضًا أن البيانات الاصطناعية لديها درجة مناسبة من الموثوقية.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Context

Currently, most cities areas are facing the problem of traffic congestion, which affects the daily work of thousands of urban residents who waste their time in traffic congestion. Therefore, this is a big problem for urban traffic and transportation planners. The administrative unit tries to solve this 2.6% problem with the help of the transportation network management system (formally known as the Intelligent Transportation System (ITS)) [1].

According to data released by the European Commission in 2017, urban areas are the engine of economic growth and employment, given that around 85% of the European Union's gross domestic product is generated in cities. Consequently, the population is increasingly concentrated in the cities. In 2010, 73% of European citizens lived in urban areas and this percentage is expected to rise to over 80% by 2050[4]. Due to the growth in population concentration, cities are faced with multiple problems related or caused by transport and traffic. One of the most important problems is the increase in the use of private vehicles and with it the saturation of urban networks. Urban traffic saturation is a problem in several ways Figure 1.1 shows the result of the article called "The hidden cost of congestion"« The hidden cost of congestion » [5]. published in Economist journal in 2018. On the left are the hours spent by the drivers of some cities in peak traffic jams in 2017. The loss of time implies a sum wasted too, the economic cost per driver is quan-

tified in the right number (in thousands of dollars). Also, the economic costs (in billions of dollars) that this represents for the cities of the United States, United Kingdom and Germany. But urban traffic does not only have temporal and economic effects according to the "World Health" Organization, exposure to ambient (outdoor) air pollution causes 4.2 million deaths each year. The same organization classifies the combustion of fuel from motor vehicles (for example, cars and heavy goods vehicles) as one of the major human activities that cause outdoor air pollution. Road accidents and noise pollution are also bad consequences of traffic, especially in urban areas. In recent years, the development of new technological solutions, the computing capacity of the systems and the amount of data generated have been increased. These factors favor the development and use of new Traffic Management Systems (TMS) and Traveler Information Systems (TIS). All of these problems can be alleviated by reducing road traffic congestion, especially in urban areas. Some of the major features of this type of system require traffic forecasts, for example Example to anticipate actions for future situations or to take navigation and route planning taking into account the future traffic state. The following section describes the traffic forecast problem and what aspects are used to differentiate its subproblems.

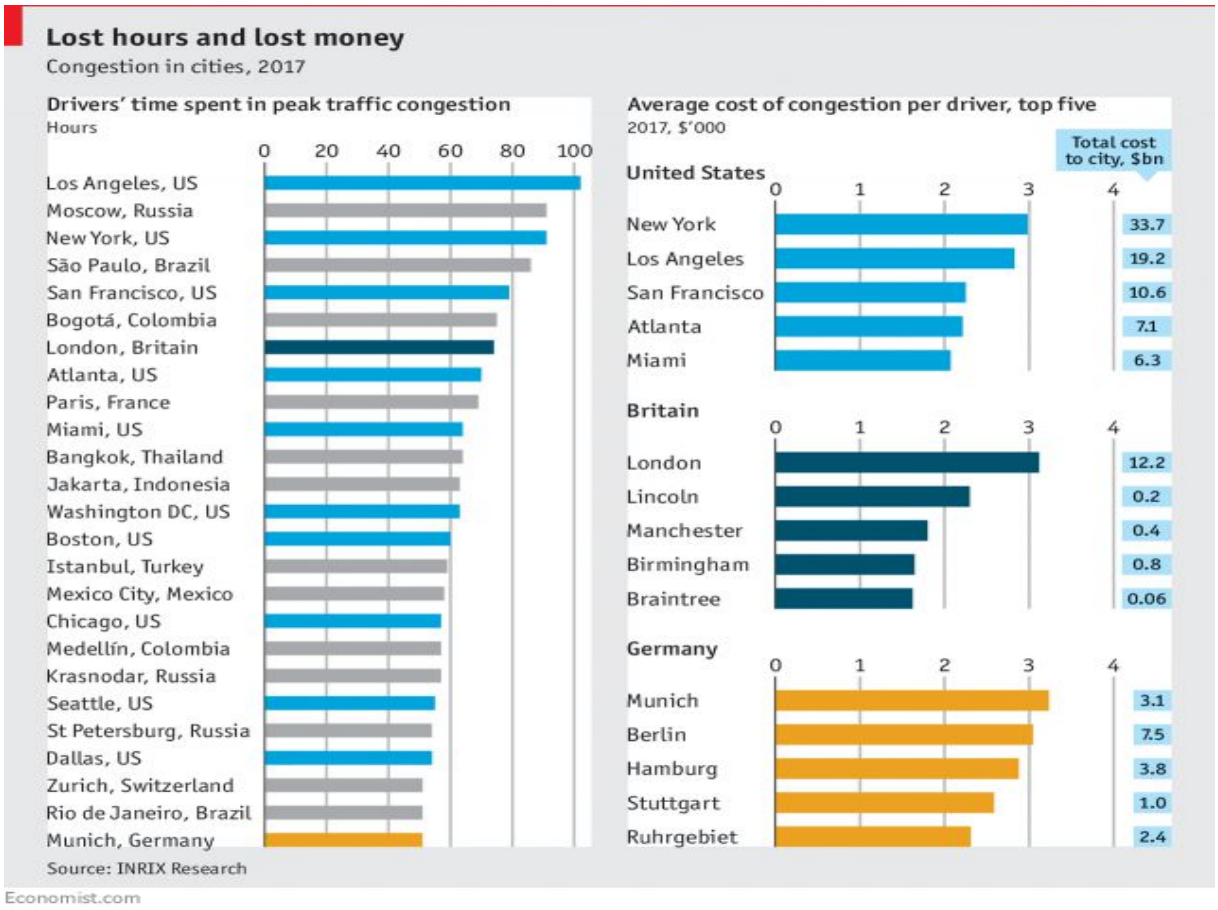

Figure 1.1: Congestion in cities 2017 [5]

1.2 The traffic forecasting problem

The problem of traffic forecasting is very broad, including several subtasks on different topics. One of the most important characteristics of a traffic forecasting task is the type of network that makes the forecast. They are usually divided into urban networks and highways. This is very different because the city network has more and shorter connections, while the highway network has fewer but larger connections. In addition, some authors only pointed out the category of the arterial network when making predictions on the main part of the city network. Generally, it is difficult to predict in urban conditions because the behavior of drivers in the network is difficult to predict. Therefore, the future of transportation will depend to a large extent on data-driven solutions to solve various problems such as traffic condition forecast for highways, data-based performance measurements and optimization of traffic parameters. control for signal synchronization,

etc.

In addition to such a classical applications For these problems such as recommanded systems ,user profiling and service offering we need reliable models that can capture the traffic flow models with better accuracy. Deep learning is one of the latest innovations in machine learning. The application of deep learning models in transport will allow us to face more complex problems and big data.

1.3 Objectives and Motivations

The main objective of this master thesis is to carry out traffic forecasts in urban contexts using two Deep Learning models formed with two different datasets real and artificial data, where the two approaches are tested, So the goal is to predict the next position or region of each car, it means whether the car will stay in the same region or not. In addition, another motivation exists for the research proposed in this master thesis for the problems of smart cities (smart city) by performing analyzes and forecasts of different traffic situations through different techniques.

The data used to test these techniques were real traffic data from the SAN FRANCISCO city network and artificial data generated with the use of the open-source SUMO (Simulation of Urban Mobility) simulator.

Another motivation for this thesis is to test if we can rely on artificial (simulated) data with a certain degree to make prediction through deep learning.

1.4 Thesis Structure

Besides this first chapter, where the context, motivation and goals of the thesis were explained, this dissertation has also the following chapters:

In chapter 2, is presented the literature review made, describing the state-of-art among genearl forecasting approaches, presenting the main existing techniques. The related works done in these subject are also presented.

In chapter 3 the implementation stage developed during the thesis is explained.

Chapter 5 covers the analyse and evaluation of the achieved results.

To finalize, chapter 5 finishes the dissertation by presenting the final remarks and future

work

CHAPTER 2

STATE OF THE ART

In this chapetr we will present briefly the concepts of deep-learning specialy the used methodes in a second secteur we will review two work that have dealed with deep learning methods to predict.

2.1 Introduction

Humans are always looking for ways to automate the different tasks and problems they encounter Against every day, this is done on the basis of there experience and professional knowledge. Computer systems are now different in terms of complexity and application areas:one can solve some type of problems with well-defined algorithms and functions that we implement and keep for the computer to execute these instructions; in other cases, this so-called traditional method cannot be used, and we prefer to introduce and deploye artificial intelligence, to be more precise, is the use of deep learning methods so that computers can solve problems like humans themselves. Given the complexity of the prediction system, it is clear that in-depth learning techniques are used more than traditional programming techniques. So what is in-depth learning technology? What technologies and methods are suitable for our case study: Vehicle Trajectory Prediction in Urban Traffic Networks ?

2.2 Deep learning

Since 2006, this machine learning class has grown strongly and has been incorporated into hundreds of research projects. Areas where in-depth learning has been integrated range from information processing to artificial intelligence. The reason for the popularity of in-depth learning can be summarised below: it has contributed to a considerable increase in the processing capabilities of computer chips, it has enabled the support of a huge amount of training data and this is the reason for the recent advances in machine learning in the field of information and signal processing.

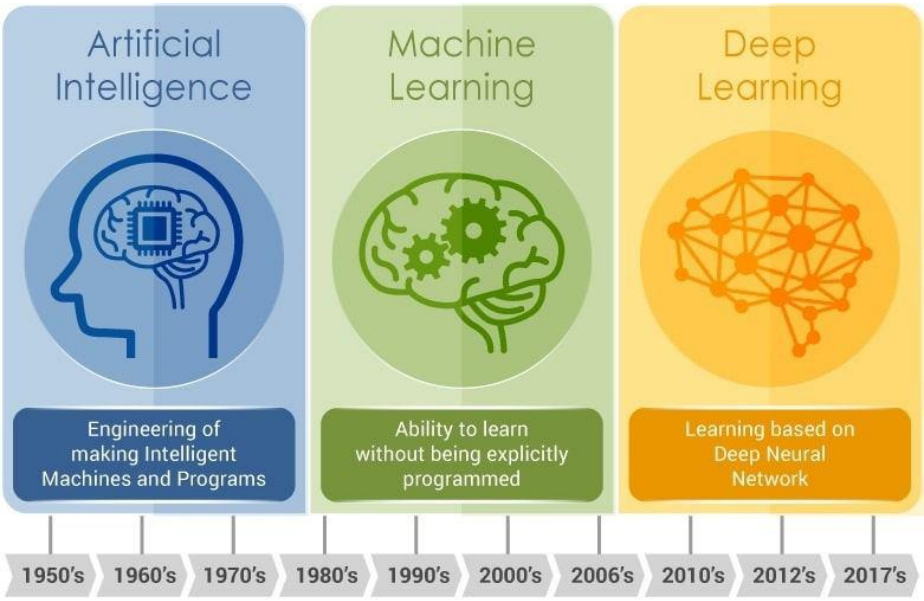

Figure 2.1: AI-ML-DL

2.2.1 Definition

Deep Learning can be described as a sub-domain of machine learning based on algorithms derived from multi-level learning in order to provide a model that represents the complex relationships between data to make the machine capable of learning by itself to train a network of neurons. It is essentially the point of intersection between neural networks, graphical modeling, optimization, artificial intelligence, pattern recognition and signal processing.

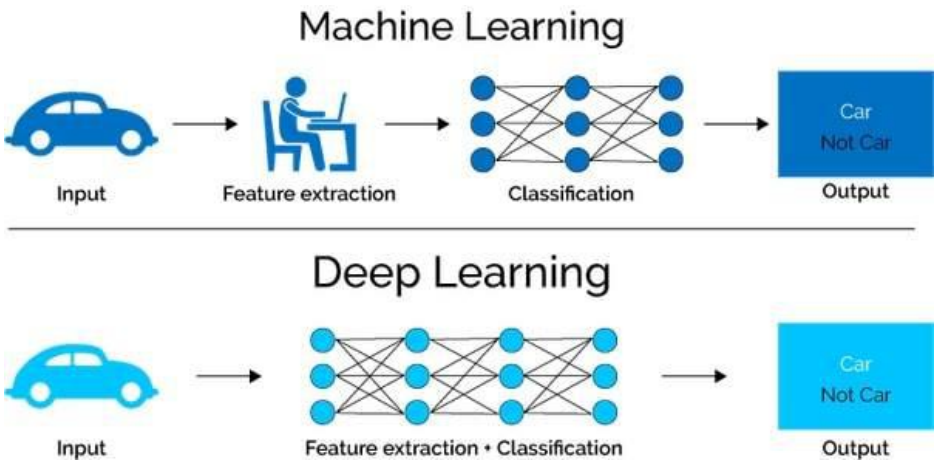

Figure 2.2: Deep learning vs machine learning [18]

Deep Learning is based on a network of deep artificial neurons inspired by the human brain. This network consists of tens or even hundreds of "layers" of neurons, each receiving and interpreting information from the previous layer. The system can, for example, recognize letters before attacking words in a text, or determine if there is a face in a photo before discovering which person it is.

2.2.2 Deep Neural Networks

The strength element of deep learning is indeed the neural networks and more precisely the networks of deep neurons. A first view of the architecture of the latter is given by the following figure:

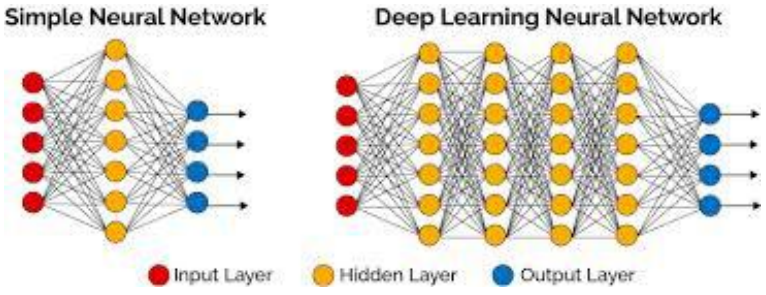

Figure 2.3: Simple Neural Network Vs Deep Neural Network [18]

2.2.2.1 Definition of Neural Network

A neural network (NN) is the association, in a more or less complex graph, of objects elementary, formal neurons. The main networks are distinguished by the organization of the graph (in layers, complete), that is to say their architecture, their level of complexity (the number of neurons, presence or not of feedback loops in the network), by the type of neurons (their transition or activation functions) and finally by the objective: supervised or unsupervised learning, optimization, dynamic systems.

Neurone formel

The biological neuron is a kind of cell in a very reductive way, which is characterized by:

- Synapses, with other neurons, nerve fibers or muscle.
- Dendritic or neuronal input
- Axon, or output from neuron to another neuron or muscle fiber
- Enable output kernel according to input stimulus

A formal neuron (also called artificial) is essentially made up of an integrator who performs the weighted sum of its inputs. The result of this sum is then transformed by a non-linear function to produce the output of the neuron. When the activation level reaches or exceeds the b threshold (b bias of the neuron, it is also called the neuron activation threshold), then the f argument obviously becomes positive (or null)

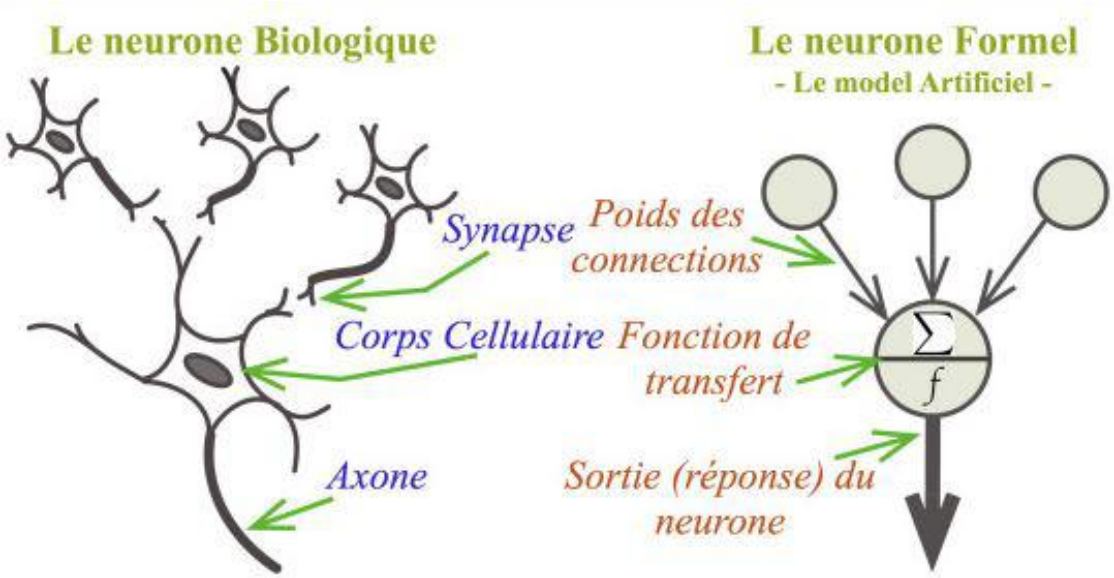

Figure 2.4: Biological Neurone VS Artificial Nurone [18]

An artificial neuron is a function f_j of $x = (x_1, ..., x_d)$ inputs weighted by a connection weight vector $Wj = (Wj,1 ,..., Wj,d)$, supplemented by neuron bj, which are associated with an activation function F , following the general formula: $Y_j = f_j(x) = F((w_j, x) + b_j)$.

Activation function

Activation functions, also known as transfer functions, are used to map input nodes to output nodes in a certain way. They perform a transformation of a refined combination of input signals weighted by a weight vector $[W_1, . . . , w_n]$ associated with each neuron, with a constant term, being called the neuron bias. The weight values are estimated in the learning phase. They constitute the distributed memory or knowledge of the network. There are several activation functions, including the most commonly used:

- identity function"id" :

  $F(x) = x$

- fonction " Segmoid " :

  $F(x) = \frac{1}{1+\exp(-x)}$

- hyperbolic tangent function "tanh" :

  $F(x) = \tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)} = \frac{\exp(2x)-1}{\exp(2x)+1}$

- fonction ”hard threshold” :

  $F(x) = 1_{x \geq \beta}$

- linear grinding unit function ”ReLU” :

  $F(x) = \max(0, x)$

The graphs corresponding to the activation functions mentioned below are shown in the following figure:

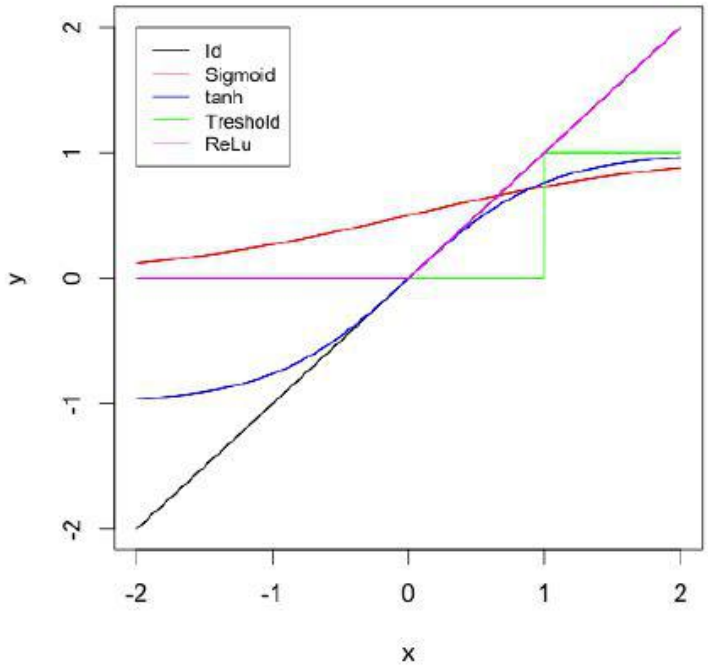

Figure 2.5: Activation functions [18]

Information enters the neural networks and passes through its layers by activating certain neurons without the others.

2.2.2.2 Types of Neural Networks

- **Convolutional Neural Networks(CNN)** are a special type of Feed Forward Networks that can process grid-like data structures more efficiently and effectively. Images can be an example of grid structured data, therefore CNNs are often used in image processing tasks. An image data can be defined as a 3D-matrix of shape

(HxWxC) where H is height, W is width and Cs are channels. Each value can be either a color value in [0, 255] or a scaled version in [0, 1] for the corresponding color channel.

Image processing can be computationally expensive when fully connected layers are used, since the number of parameters will be large. CNNs handle this problem by reducing the number of parameters that will be trained with shared parameters for the layers. These parameters are called kernel that is a matrix used in convolution operation over the input[6].

NN is commonly used in visual feature extraction tasks since they have the ability to capture edges or common characteristics in the data. In this way, a neural network can be trained to extract the crucial features to solve a task. For example, this can be ear and nose shapes in a cat and dog detection task. Besides, in the best practice, CNN is used with activation functions and pooling operations that is reviewed in the next section in more detail.

Convolution Operation (cnn)

The convolution operation is achieved by moving the kernel along horizontal and vertical directions of the grid structured data/image. While the kernel is moving, all pixels that are covered by the kernel in the image are multiplied element-wise with the kernel pixels and the resulting values are summed. In this way, one pixel of the resulting matrix is obtained. The step length of the kernel is controlled by the stride parameter. For instance, stride=1 means that moving the kernel by 1 pixel. This operation is visualizes in Figure 2.6. Finally, the calculated output is a filtered version of the input and the type of visual feature that is captured depends on the values of the kernel elements. The training is done to learn extracting useful visual features to solve the target task.

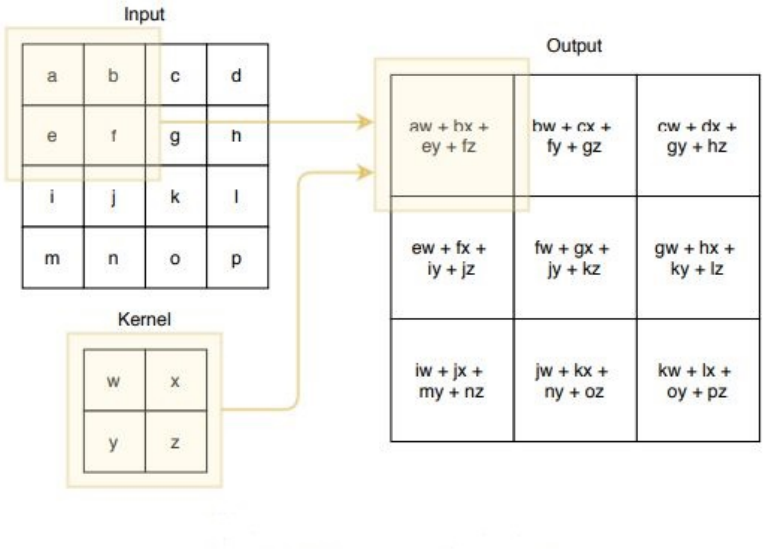

Figure 2.6: convolutions operations [18]

Pooling Operation

Pooling functions apply a summary statistics function to a rectangular neighborhood. One of the common pooling functions is the max pooling function that returns the maximum output that is within the rectangular neighborhood Figure 2.7 shows how pooling is applied with a 2x2 filter and stride=2. As can be seen in the Figure, the size of the image is reduced. By the help of the stride, the input is downsized with respect to height and width [7]. This is a very common method to apply down sampling in order to capture meaningful and more complex features in a CNN.

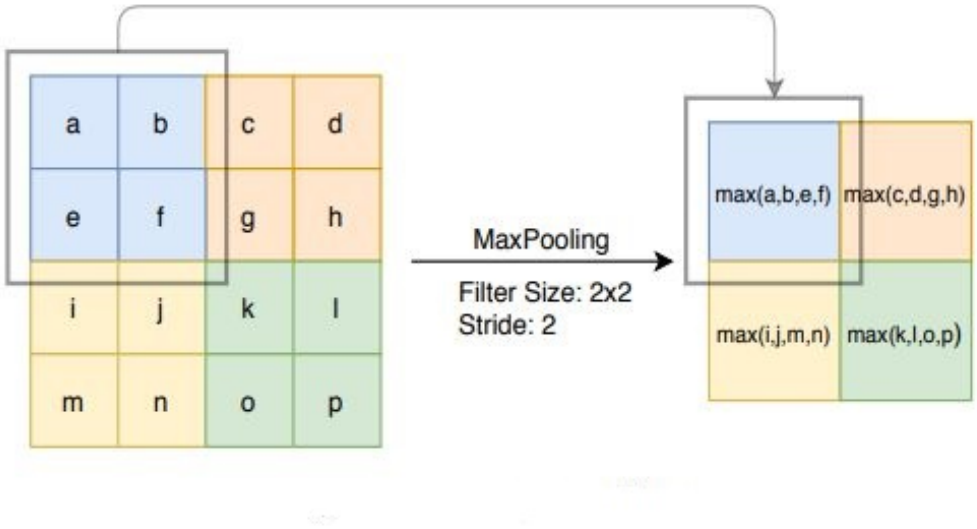

Figure 2.7: Pooling Operation example

- **Recurrent Neural Networks** A network whose directional graph does not contain cycles is considered a forward neural network (this is the traditional neural network method). In contrast, a network whose graph contains cycles is called a Recurrent Neural Network (RNN). Although this seems to be a completely different solution, when unrolling the loop, you can think of the RNN as multiple copies of the same network, and each copy sends a message to its successor. Figure 2.8 shows the expanded RNN graph, where A represents the repeated part of the NN, and x_t is the input that produces the value x_t. Contrary to the human thought process, feed-forward Neural Networks start each process from scratch. Many channels and list tasks can use the previous state to improve the next result. Therefore, RNN enables information to flow from one network hop to the next, while the information remains between entities. Some examples of using these models are based on time series problems, where repeated feedback uses time relationships between data.

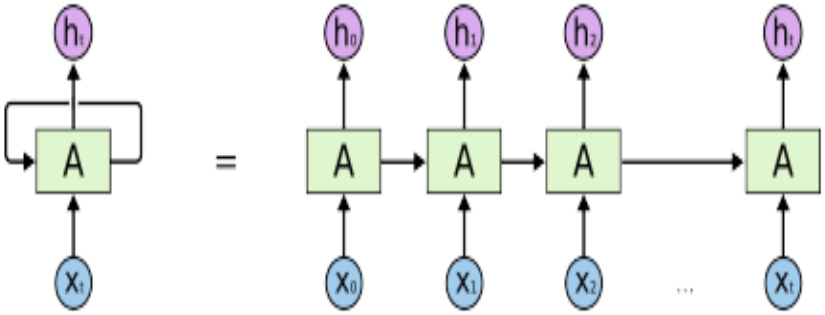

Figure 2.8: Schema of an unrolled Recurrent Neural Network

Long Short-Term Memory Neural Networks In theory, simple RNNs can handle "long-term dependencies", but in reality they cannot learn. To solve this problem, Hochreiter and Schmidhuber [1997]proposed a short-term long-term memory neural network (LSTM), which is a type of RNN that can store information for a long time. The difference between simple RNN and LSTM is The structure of the repeating unit In a standard RNN, the structure of this repeating unit is very simple, for example: Hyperbolic tangent layer; but as shown in Figure 2.9, in LSTM, there are four, instead of a first-level neural network. In order to better understand these figures, the symbols shown in Figure 2.10 are used.

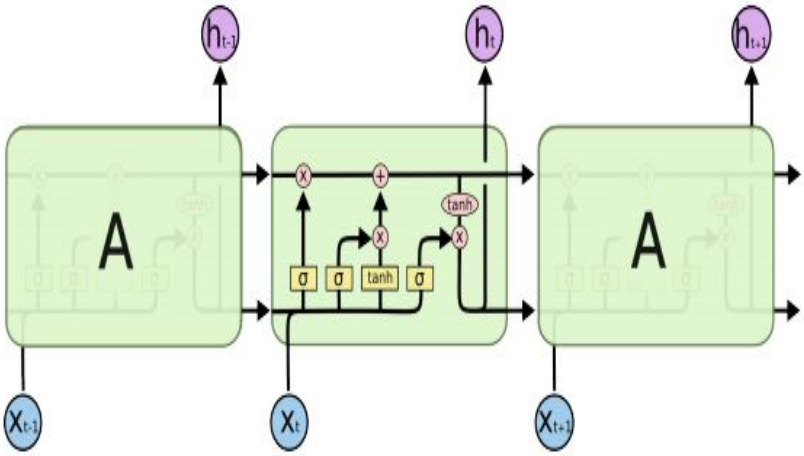

Figure 2.9: Schema of an unrolled Recurrent Neural Network

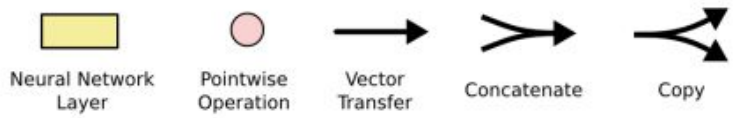

Figure 2.10: Notation of the LSTM schemas.

The main feature of LSTM is the double forwarding connection in the repeater module. In addition to the typical recurrent connections in RNNs that reuse previously predicted outputs, LSTMs also have cell states (represented by the top horizontal arrow in Figure 2.10). This state of the unit allows the maintenance of accumulated information, which can be changed after some minor linear interactions. The internal behavior of LSTM can be explained in 4 steps:

1-The first part of the LSTM is called the "forget gate" (Figure 2.11), which determines what information to remove from the cell state. A sigmoid layer is used that combines h_{t-1}. (hidden state) and x^t. (input) and outputs a number between 0 (completely deleted) and 1 (completely reserved) in the cell state c_{t-1} .

$$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f) \qquad \text{(Equation 3.2)}$$

2-the second one, named input gate (Figure 2.12), selects what information has to be added to the cell state. It is formed by two parts. First, a sigmoid layer uses h_{t-1} and x_t and then, i_t generates a vector i_t of values between 0 and 1 (Equation 3.3). These values quantify what portion of the state values will be updated. Next, a tanh layer combines h_{t-1} and x_t to create a vector of new candidate values, Ct

(Equation 3.4)

$$i_t = F(W_i[x_t, h_{t-1}] + b_i) \qquad \text{(Equation3.3)}$$

$$c_t = \tanh(W_c[x_t, h_{t-1}] + b_c) \qquad \text{(Equation3.4)}$$

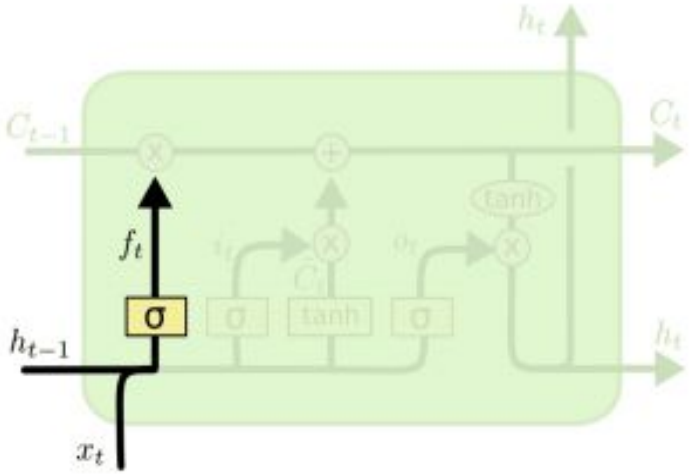

Figure 2.11: Schema of an unrolled Recurrent Neural Network (step1)

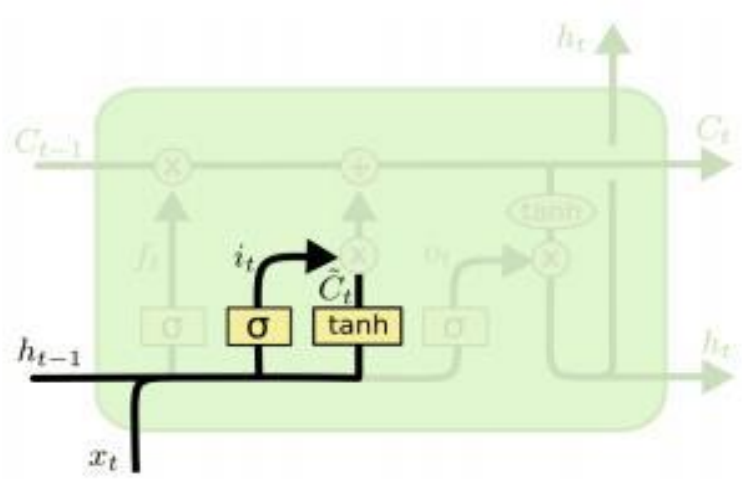

Figure 2.12: Schema of an unrolled Recurrent Neural Network (step2)

3-Based on the results of the previous steps, the cell state can be updated (Figure2.12). By multiplying the previous state by f_t , the new cell state will forget the information selected in the first step, so the new information is added to the new information in step 2. This new information is obtained by multiply the i_t (which values have to be updated) and c_t (the new candidates values). The full operation is in Equation 3.5

$$c_t = f_t \times c_{t-1} + i_t \times c_t \qquad \text{(Equation 3.5)}$$

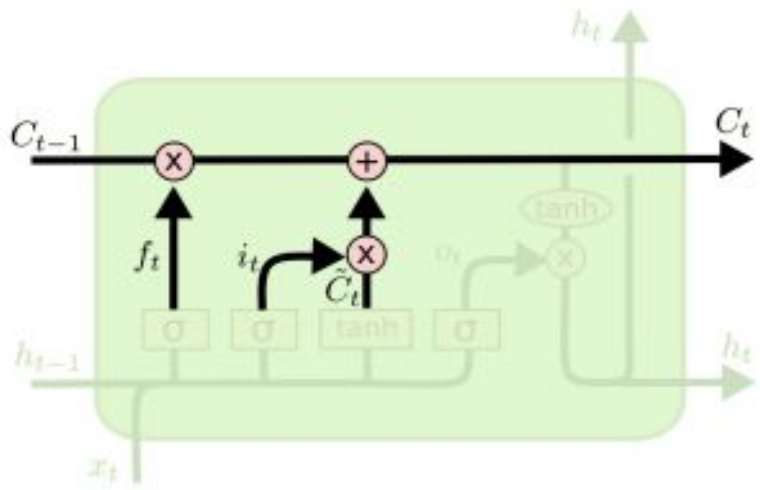

Figure 2.13: Schema of an unrolled Recurrent Neural Network (step3)

4-The last step (Figure 2.13) is to compute the output, which is a filtered version of the updated cell state. First, a sigmoid layer decides, based in h_{t-1} and x_t, what parts of the cell state are relevant for the output (Equation 3.6). Then, the cell state is mapped to values between -1 and 1 using a tanh multiplied by the results from the sigmoid layer (Equation 3.7)

$$o_t = F(W_{x0}[x_t, h_{t-1} + b_0]) \qquad \text{(Equation 3.6)}$$

$$h_t = o_t \times \tanh(C_t) \qquad \text{(Equation 3.7).}$$

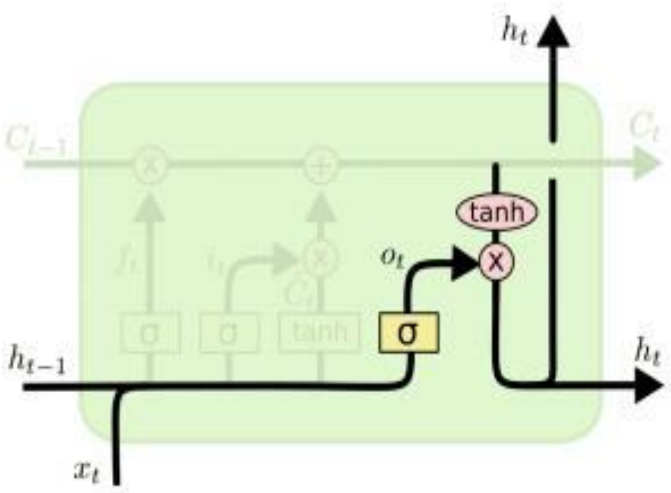

Figure 2.14: Schema of an unrolled Recurrent Neural Network(step4)

2.3 Forecasting Approaches

Research on methods of forecasting traffic volume in the short term has always been one of the centers of research. In the 1960s and 1970s, some researchers began to apply Mature Forecasting Methods in Economics, Physics, and other discipline for short-term traffic flow forecasting. Forecasting methods are mainly using in linear and statistical theories. With the application of the advanced work Smart Short-Term Traffic Flow Prediction Algorithm prediction accuracy has been improved with a certain method. Currently, short-term traffic forecasting methods can be divided into five categories: Statistical analysis Model, artificial intelligence model, nonlinear theory, transport Combination of simulation and prediction models [26].

2.3.1 Statistical Analysis Model

Statistical analysis model based on the short-term traffic flow prediction method. This method is based on the characteristics of the time series of short-term traffic flows, and use statistical analysis models with greater applicability adapt to the trend of the traffic flow, in order to realize the flow prediction in short-term . The application of this method is Mature, including historical mean model, serial model Kalman filter theory and nonparametric regression model [12]

2.3.2 Non linear Theory

This method builds the corresponding prediction method using the theories of a nonlinear system like chaos theory, dissipative structure theory and theory of self-organization. At present, relatively mature prediction methods include wavelet theory, mutation theory, and chaos theory [15]

2.3.3 Short-term traffic prediction based on traffic simulation Method

Short-term traffic prediction based on the traffic simulation method. This method takes the vehicle as an entity, describes the traffic infrastructure of the road network and the traffic behavior of drivers on the road network with relevant models and algorithms, and combines the traffic flow model with computer microsimulation technology to Dynamically simulate the state of vehicle traffic on the road network, so as to predict short-term traffic flow data. The main theoretical research results of this method are as follows: a method of using cellular model to predict the flow of traffic in the large-scale traffic network is discussed; Micro-traffic simulation technology is used to predict short-term traffic flow on the fast highway, microcosmic traffic simulation technology is used to predict the flow of traffic on the freeway. Based on the idea of dynamic programming, a dynamic method of predicting the flow of traffic on the road network is proposed

2.3.4 Short-term traffic flow prediction based on prediction model

Short-term traffic flow prediction based on prediction models. This method uses at least two forecasting methods to simultaneously predict short-term traffic flow in order to take full advantage of the advantages of different forecasting methods. Main combination model of this method are: neural network model and ARIMA model[1], RBF neural net-

[1]ARIMA: short for 'AutoRegressive Integrated Moving Average', is a forecasting algorithm based on the idea that the information in the past values of the time series can alone be used to predict the future values.

work model [2] and fuzzy mean value, neural network model and fuzzy decision system, model neural network and genetic algorithm, algorithm of ant colony and vector machine support (SVM), Kalman filtering and secondary index of the noise reduction method, the Kalman filtering model and artificial neural network model and the full fuzzy model, Back Propagation (BP) network model and ARIMA model with wavelet analysis

2.3.5 Artificial Intelligence Model

This method combines time series uncontrollability of the short-term traffic flow intelligence model as a training method, and then generate predicted value of the traffic flow. Artificial Intelligence Model Mainly include Neural Network (NN) Model, Vector Machine (SVM) etc..[13] [14]

Related to our proposed approach we have chosen to review 2 works

2.3.5.1 DeepVM: RNN-based Vehicle Mobility Prediction to Support Intelligent Vehicle Applications

The article objective is to apply deep learning technology to predict the micro-level mobility of a vehicle by using its mobility trajectory directly.

The authors proposes an algorithm to predict the probability for a vehicle to enter any city region in a concerned future time when its mobility trajectory is given, assumed that vehicles move in a city divided into grids. The size of a grid is from some hundred meters to very few kilometers, The time is also divided into a sequence of time slots. In every time slot, vehicles save their positions through a positioning system such as GPS.

The DEEPVM algorithm uses a deep RNN architecture and processes a 16-order vehicular trajectory to predict vehicle mobility.

Figure 2.15 shows the neural network architecture of DeepVM unfolded in time.

[2] RBF neural network model:In the field of mathematical modeling, a radial basis function network is an artificial neural network that uses radial basis functions as activation functions. The output of the network is a linear combination of radial basis functions of the inputs and neuron parameters.

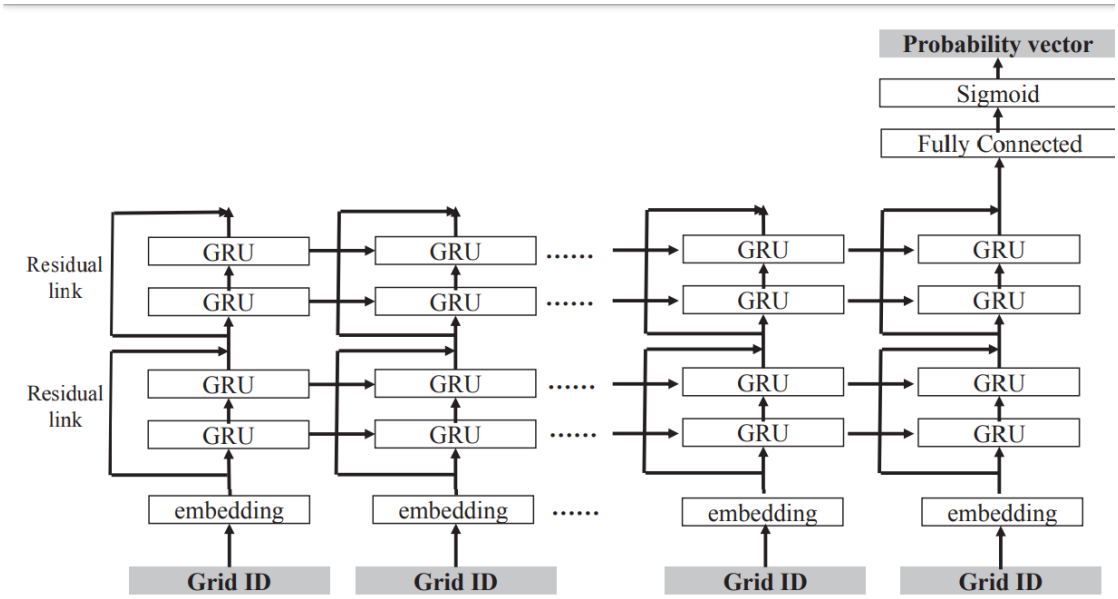

Figure 2.15: The neural network architecture of DeepVM algorithm unfolded in time.

DeepVM first encodes every grid identity to an n-dimensional one-hot vector where n is the number of grids in city space, e.g., a grid identity of 2 is encoded to (0, 0, 1, 0 · · · 0). Every input data of DeepVM includes a sequence of 16 grid identities that shows us the location for a vehicle in the past 16 time slots. In consequence, DeepVM uses an embedding layer to transform a sparse grid identity vector into a smaller- and-denser feature vector.

The embedded vector of vehicular trajectory is fed into the RNN cells of DeepVM, DeepVM combines two GRU[3] blocks, and each block is composed of two layers of GRU cells. When trained by the embedded vectors of vehicular trajectory, these GRU blocks are able to learn and store the spatial-temporal correlations among the embedded vectors and use them to make predictions. A residual link is added to connect the input and output.

in the last A fully connected layer with sigmoid activation transforms the output vector of the last GRU cell to a real-valued probability vector.

Thier models has been generated using dataset based on the real taxi mobility data that are collected from a wireless IoT testbed located in Tokyo, Japan,this data set contains

[3]GRU:Introduced by Cho, et al. in 2014, GRU (Gated Recurrent Unit) aims to solve the vanishing gradient problem which comes with a standard recurrent neural network

the mobility data of 65 taxis in a period of 4 months.[36]

2.3.5.2 Edge-Assisted Vehicle Mobility Prediction to Support V2X Communications

the paper proposed algorithm uses a hybrid architecture of convolutional and recurrent neural networks, and allows computationally efficient transfer learning in each vehicle to generate its customized mobility prediction model. Extensive evaluations have been conducted by using a real taxi mobility data set that is collected from a testbed deployed in Tokyo, Japan. The results have validated that, compared with other state-of-art algorithms, their proposal improves the prediction F1 score of vehicle mobility by more than 30%, especially for those vehicles that own a strong individual mobility preference.

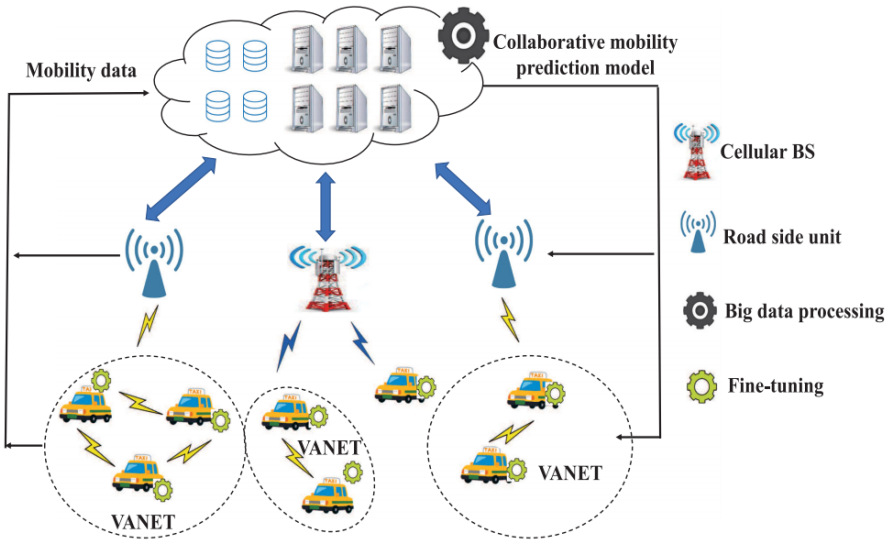

Figure 2.16: The neural network architecture of DeepVM algorithm unfolded in time.

Figure 2.16 represents an overview of the proposed algorithm framework. It is assumed that several vehicles move in a geographical space like a city, and they regularly save their instant mobility data by using any positioning system like GPS. These mobility data are not only kept in its own storage, but also sent to a data center by either the cellular communications between vehicles and data center or the DSRC [4] communications

[4]DSRC:Dedicated Short Range Communications (DSRC) is an open-source protocol for wireless communication, similar in some respects to WiFi. While WiFi is used mainly for wireless Local Area Net-

between vehicles and RSUs. Obviously, these massive and historical vehicle mobility data accumulated in the data center contains the mobility patterns of many different vehicles in the concerned space. in consequence, these data are fed to a deep neural network to train a prediction model that infers vehicle mobility by referencing the mobility patterns of several vehicles collaboratively. In addition, this sort of collaborative prediction model is trained with the large mobility data of many vehicles in a relative long historical period like a few months or years.

However, since the training process of collaborative prediction model does not identify the mobility data from different vehicles separately, it may fail to identify the individual mobility preference of each single vehicle. So, thier proposed EVM algorithm lets each vehicle customize a prediction model by using its own mobility data to fine-tune the received collaborative prediction model.[35]

The following table summerizes the main features of the reviewed of related work those have used Deep learning methodes

Work	Dataset	Predection Period	Architecture	F1 score of 1 min
Deep VM	Dataset of 64 taxi in tokyo japan for 4 months	1min , 5 min , 10 min ,20 min	RNN and GRU	0.69
Edge-Assisted	Dataset of 64 taxi in tokyo japan for 4 months	1min , 5min , 10 min , 20 min	Hybride :CNN + RNN	0.7

works, DSRC is intended for highly secure, high-speed wireless communication between vehicles and the infrastructure

CHAPTER 3

OUR APPROACH

This chapter begins by providing a detailed description of the problem and then exploring the analyzes performed to identify the needs. Next, the proposed solution for implementing model and data-based forecasting is presented, presenting the methodology and how to do it. In the end an overview of the technologies to be used is presented.

3.1 Problem Statement

Real-time traffic forecasting on road networks has acquired great importance in current ITS, due to the fact that if there is a chance to be able to predict, with some degree of confidence, the state of the network in a near future which would lead to an improvement in the management performance of road networks. Basically, the existing techniques for implementing traffic forecasts are through the use of simulations or data-driven approaches, also the problem of data scarcity is very important since data are at the core of any AI project. the size of a dataset is often responsible for poor performances in ML projects, most of the time, data related issues are the main reason why great AI projects cannot be accomplished. In some projects, you come to the conclusion that there is no relevant data or the collection process is too difficult and time-consuming. this work aims to develop and test two different approches through simulation and real mesured data to test if simulation techniques can be relied on using two Deep learning models.
The idea behind that is to respond to the following questions :
In the case of lack of real data can we rely on artificial ones to train accurate trafic

forcasting models ?

3.2 General Overview

Figure 3.1 illustrate our investigation in fact we aims to develop and test certain techniques resulting from these two different approaches, in order to analyze the advantages of the application of traffic forecasts on ITS. Therefore, the simulation technique chosen was the microscopic approach because, despite the high computational consumption that it might require, it is the approach that allows to simulate the traffic and its behavior in a more realistic way. Regarding data-based approaches, the option made was to use artificial neural networks (CNN) and Long short-term memory (LSTM) as they are models that can be trained and "designed" according to real data, and when properly trained, it has a high level of reliability. To conclude, the two approaches will be tested using real and artificial data and their performance will be compared.

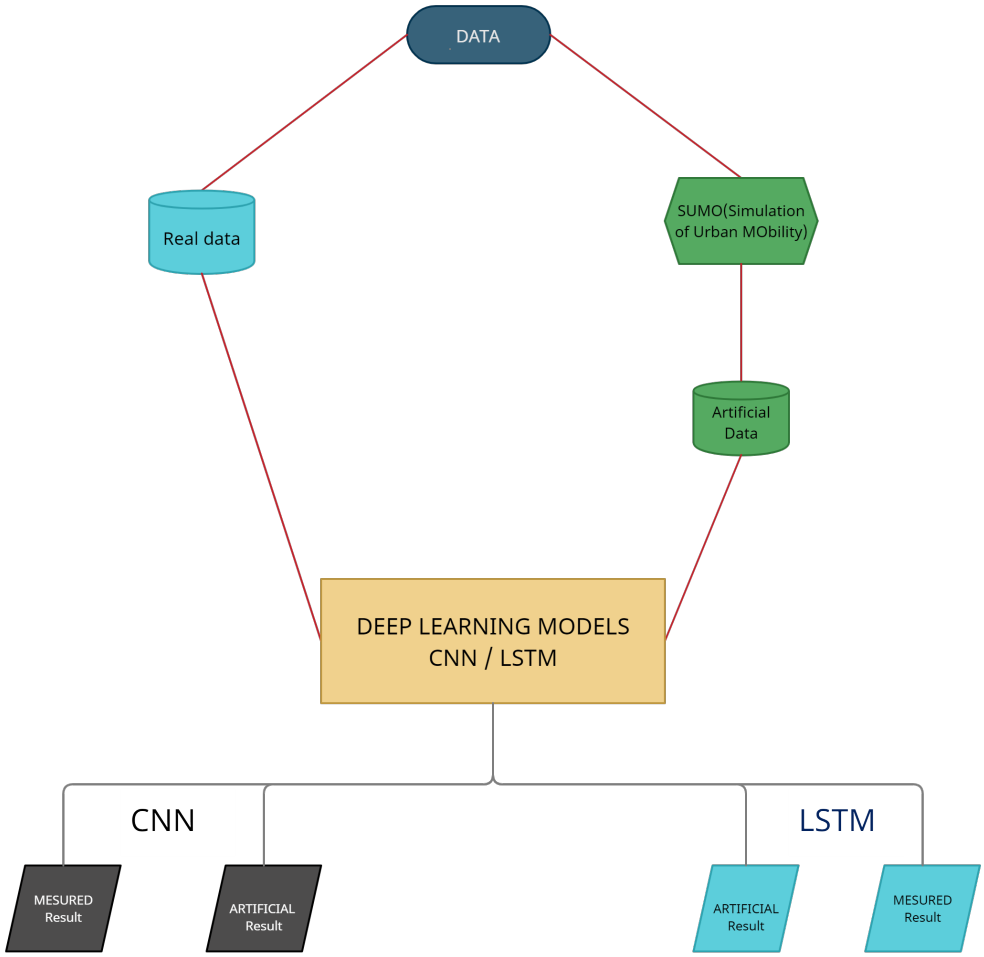

Figure 3.1: Overview of the system

3.3 Analyze of needs

Before starting the implementation, it was necessary to identify the technological requirements for this investigation a empirical . Starting from traffic simulation, the traffic simulator chosen was SUMO (Simulation of Urban MObility), which is one of the most used microscopic simulators in the research community, and the fact that it is lightweight and open-source made it the best option. The road network used was the city of SAN FRANCISCO and a more detailed description of it can be found in the dataset section. To develop the artificial neural network, it was decided to use the Keras Framework This is a high-level neural network API, written in Python. Although it is capable of running on different backends, and Jupyter Notebook who is a web based application which makes

us able to create and modify live codes, equations, plaintexts and visualizations.

3.4 Experimentation

In this section, all methods and tools used to make the vehicles trajectory prediction are being discussed. At the very beginning, tools that are utilized in the research is described in details. After that, the implementation process started with data generation, preprocessing and models are presented with a short summary at the end of the section

3.4.1 Used Tools

Like every other researches and studies this work utilized some applications and tools for creating models and performing experiments. For such researches many tools and libraries which are necessary or we can say useful in order to create the models.

3.4.1.1 Keras

Keras is the most used deep learning framework and it is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent and it minimizes the number of user actions required for common use cases, and it provides clear and actionable error messages. It also has extensive documentation and developer guides.
Keras has the low level flexibility to implement arbitrary research ideas while offering optional high level convenience features to speed up experimentation cycles.

3.4.1.2 Jupyter Notebook

Jupyter Notebook is a web based application which makes us able to create and modify live codes, equations, plaintexts and visualizations. This is an open source notebook which supports many programming languages. This is used for different purposes like machine learning, numerical simulation, information visualization etc.

3.4.1.3 Python

Python is a general-purpose programming language developed by Guido van Rossum, which can be used on multiple platforms, such as mathematics, computer graphics, the Internet, and many scientific applications. The main purpose of creating Python is to make it easier for everyone in the world to write code. Therefore, it is popular for its simplicity. Python is sensitive to space. Although Python is designed for children, it currently integrates all programming languages in many areas, which is amazing and incredible. In our work we deployed the following libraries.

- **Numpy** Numpy is an open source library, it does the computing with the help of multi-dimensional matrices and arrays. It contains a number of functions which makes it easy to work with our these type of data. In data analysis if we want to make the speed fast and efficient we need to use arrays, therefore this library helps to work faster with large amounts of data. Usually for detection and forecasting, the models function needs arrays as parameter to operate fast and decrease the training prediction time. Matplotlib

- **Pandas** is a Python package that provides fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language.

- **Matplotlib** Plotting is growing in all fields to visualize the data and to make it understandable. Therefore Matplotlib is being used as a plotting library to create different kind of graphs and figures for variety of aims. The good thing about matplotlib is that it can produce good plots and graphs with just few lines of codes. So matplotlib is used to extract color features and create histograms

- **contextily** is a small Python 3 (3.6 and above) package to retrieve tile maps from the internet. It can add those tiles as basemap to matplotlib figures or write tile maps to disk into geospatial raster files. Bounding boxes can be passed in both WGS84 (EPSG:4326) and Spheric Mercator (EPSG:3857)

- **Sklearn** is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, is built upon NumPy, SciPy and Matplotlib.

- **SeaBorn** is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

3.4.2 Datasets

For the datasets we have two datasets: one is realDataset and the other artificial dataset generated using sumo (urban of mobily) for the city of San Francisco ,Both datasets contain 20 traces of vehicles for 1 week in which the total vehicules observation for artificial dataset is 117 022 and 46718 for realDataset .

3.4.2.1 Real dataset

This real dataset are the traces of mobility provided by the Exploratorium - the museum of science, art and human perception through the CabsPotting project.
Each San Francisco-based Yellow Cab vehicle is currently equipped with a GPS tracking device that is used by dispatchers to effectively reach customers. The data is transmitted from each cab to a central reception station and delivered in real time to dispatch computers via a central server.This system broadcasts the taxi's call number, its location and whether the taxi currently has a fare.
. This archive contains the file cabs.txt with the list of all cabs and for each cab its mobility trace in a separate ASCII file, for example 'newabboip.txt The format of each mobility trace file is as follows - each line contains [latitude, longitude, occupation, time], for example: [37.75134 -122.39488 0 1213084687], where latitude and longitude are in decimal degrees, occupation indicates whether a taxi has a fare (1 = occupied, 0 = free) and time is in UNIX period format.
Figure3.2 illustrates the content of such file traces .

_time	vehicle._id	vehicle._y	vehicle._x
29793.0	45	-122.452969	37.775027
29794.0	45	-122.453077	37.775013
29795.0	45	-122.453206	37.774997
29796.0	45	-122.453360	37.774977
29797.0	45	-122.453526	37.774956
...	...	...	...
590213.0	45	-122.403608	37.738928
590214.0	45	-122.403751	37.738888
590215.0	45	-122.403888	37.738851
590216.0	45	-122.404019	37.738815
590217.0	45	-122.404159	37.738777

Figure 3.2: A Simple for real dataset

3.4.2.2 Artificial Dataset Generation

Simulation of Urban MObility (SUMO) Technology Overview: SUMO is an open-source microscopic traffic simulator created by the German Aerospace Center (DLR) in 2001, with the objective of providing a tool that would be fast in operating time and portable, as well capable of supporting new algorithms developed by its users It is currently in the version 0.20 and it has been widely used by the traffic simulation community. An important feature is the fact that SUMO is a multimodal simulator, capable not only of representing the movements of cars, but also the other type of vehicles that usually circulate in a city, such as motorcycles, bicycles or pedestrians, as well as public transport vehicles: bus, train, tram or taxi. The simulator is available on the command line or via a graphical user interface .

The format of the files used by SUMO is based on XML, depending on the extension used depending on the file type. For example, the simulation configuration file is saved with a .sumo.cfg extension; the network is defined in a file using the .net.xml extension; or the routes file using the .rou.xml extension. Besides the simulator, SUMO also has a set of small applications included in its package that work as auxiliary tools for tasks such as: generating the network to use (netconvert, netgenerate), define and calculate the vehicle routes (darouter, jtrrouter, dfrouter), by entering the vehicle demand in the simulation (od2trips), as well as for other uses (e.g.: polyconvert, edgeinDistrict).

Data generation

Step1: -the first step of the implementation was to convert the map of the San Francisco area in OpenStreetMap format into a readable format for SUMO. This conversion was made with SUMO application netconvert which allows to read a card from the . osm format and to convert it into a format of the network file used by SUMO (.net.xml)

```
netconvert --osm-files sanfrancisco.osm -o test1.net.xml --plain-output-prefix plain --proj.plain-geo
```

Step2: -After generating a network, you could watch it using SUMO-GUI, but no cars would be in the network. We still need a kind of description of the vehicles and how they circulate in the topology of the network. This is called traffic demand.
In order to understand the traffic demand, the following terminology is required. A journey is a movement of a vehicle (or persons) from one place to another defined by the departure edge (street), the destination edge and the departure time. An itinerary is a journey, which means that an itinerary definition contains not only the first and last segment, but the edges crossed by the vehicle, from its origin to its final destination.

Step3: -Then specify random traffic with the Python randomTrips.py script (available in the directory tools in SUMO) to generate a set of random paths for a given network, which will choose a source and a destination edge is uniformly random, or with a spicific Distribution. The resulting routes are stored in an XML file adapted to DUAROUTER that can be called automatically by the script with the right options enabled. Trips are distributed uniformly in an interval defined by configurable start and end times. the ligne commande to do such generation is :

```
python "/usr/share/sumo/tools/randomTrips.py" -n test1.net.xml -r test2.rou.xml -e 500  -l
```

-this command allows to generate random trips representing in .rou.xml and trips.xml

with takes as input the file extension .net.xml which contains the map of San Francisco. "randomTrips.py" generates a set of random paths for a given network (option -n). To do this, it chooses the source and destination edges either uniformly at random, or with a modified distribution as described below. The resulting routes are stored in an XML file (option -o, by default trips.trips.xml) adapted to DUAROUTER which is called automatically if the option (with a file name for the resulting route file) is given. The trips are evenly distributed over an interval defined by the start time (option -b, default 0) and the end time (option -e, default 3600) in seconds. The number of trips is defined by the repetition rate (option -p, default 1) in seconds. Each trip has an identifier consisting of a prefix (option –prefix, default "") and a number in progress. -then launches the simulation and to generate the trace file use the FCD (Floating Car Data) export which contains the location and speed as well as other information for each vehicle in the network at each time step. The output behaves a bit like an ultra-precise high-frequency GPS device for each vehicle. Outputs can be processed later using the TraceExporter tool to adapt the frequency, equipment rates, accuracy and data format

```
sumo -c myconfig.sumocfg --fcd-output.geo true --fcd-output  sumoTrace1.xml
```

-Note that SUMO networks are always coded in Cartesian coordinates (meters) and may contain geo-referencing information to allow conversion to longitude and latitude. By default, the cartesian coordinates use the UTM [1] projection with the origin shifted to so that the lower left corner of the network is 0.0. therefore it is necessary to ensure that the simulator uses the geo-coordinates (longitude and latitude) -the next step of doing the trace filtering and getting the data needed to make the prediction , the position x ,y and timestamp and mastering the trace of each car in a separate csv file for in-depth learning

[1] UTM:Universal Transverse of Mercator

3.4.3 Used metrics

Evaluation metrics is a key factor in assessing performance of classifier. In two-class problem, confusion matrix is used to describe classifier predictive performance. Confusion matrix is a twodimensional table that visualizes correctly or incorrectly predicted sample of each class by matching the actual and predicted value of all samples. It is especially useful in supervised learning, where each sample is labelled with a target class. Figure 2.6 shows an example of confusion matrix of two-class problem. In this example, actual class refers to the actual target class of an input data which is taken as the ground truth, while predicted class refers to classifier output of the same input data. In each dimension, label P denotes positive class and N denotes negative class.

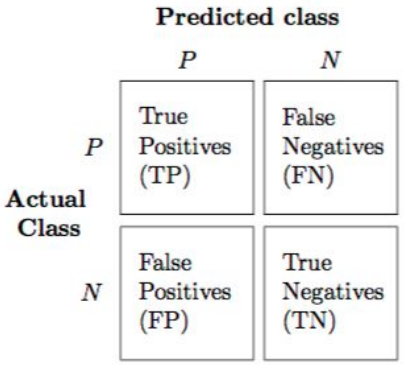

Figure 3.3: Confusion Matrix of two-class problem

When prediction outcome of a classifier matches the actual class, we call such case True Positive if the actual and predicted class is positive, True Negative when actual and predicted class is negative. In case a classifier incorrectly predict negative class as positive or incorrectly predict positive class as negative Once confusion matrix is constructed, several other metrics can be derived from it, namely:

Accuracy

Accuracy is the most common metrics used in data mining studies. It is simply defined as the proportion of correct decision made and total number of decisions made. However, accuracy does not always work well in all cases. In data with imbalance class, accuracy is no longer a proper measure because it could be biased to majority class and does

not reflect the performance of minority class. It is possible that a classifier have a high accuracy by predicting all samples as negative case.
Accuracy = $\frac{TP+TN}{TP+FP+FN+TN}$

Precision

Precision (or also called Positive Predictive Value) is defined as proportion of positive case that are correctly identified in outcomes.
Precision = $\frac{TP}{TP+FP}$

Recall

Recall, also known as sensitivity or true positive rate (TPR), is a measure of the ratio of the true observations that the model will classify as true. Recall = $\frac{TP}{TP+FN}$ Recall answer the question: "Given a positive example, will the classifier detect it?" The recall values go from 0, indicating zero true positives, to 1, indicating zero false negatives. A low recall indicates many false negatives.

F1 Score

F1 score is defined as weighted average of Precision and Recall
F1 = $2.\frac{Precision.Recall}{Precision+Recall}$

3.4.4 Implementation

In this section some details on the approche are reported.

3.4.4.1 Processing the time index

The measured data are recorded with an irregular time step, so the first treatment basic data is to regularize the recording time. The time step recording time chosen will be 1 min. If in a given time, the measurement is non-existent, the index of the observations will be created but no value will be assigned to the corresponding columns (latitude, longitude). For artificial data, the measurements do not have a time index, therefore a time index artificial will be assigned to them, the date starting from January 1, 2020 at midnight, will continue until the end of the series of each cab.

3.4.4.2 Dividinng the Map into grids

In order to simplify the problem, the result of the prediction will not be the precise GPS position, but the area or region where the vehicle is located. We will first start by meshing the activity area of each taxi, the mesh chosen is a rectangular mesh that depends on the amplitude of the GPS coordinates. The mesh extends area extends from the max and min position of latitude and longitude, plus a coefficient that allows to significantly extend the area, the value of this coefficient is calculated by multiplying 20% by the amplitude of each coordinate (amplitude = max - min).

In order to mesh the area we need to divide the mesh boundaries by an integer, in our case we will choose number of division = 20.

Each measured or simulated observation will be assigned an integer value representing the identifier of the cell where the vehicle at.

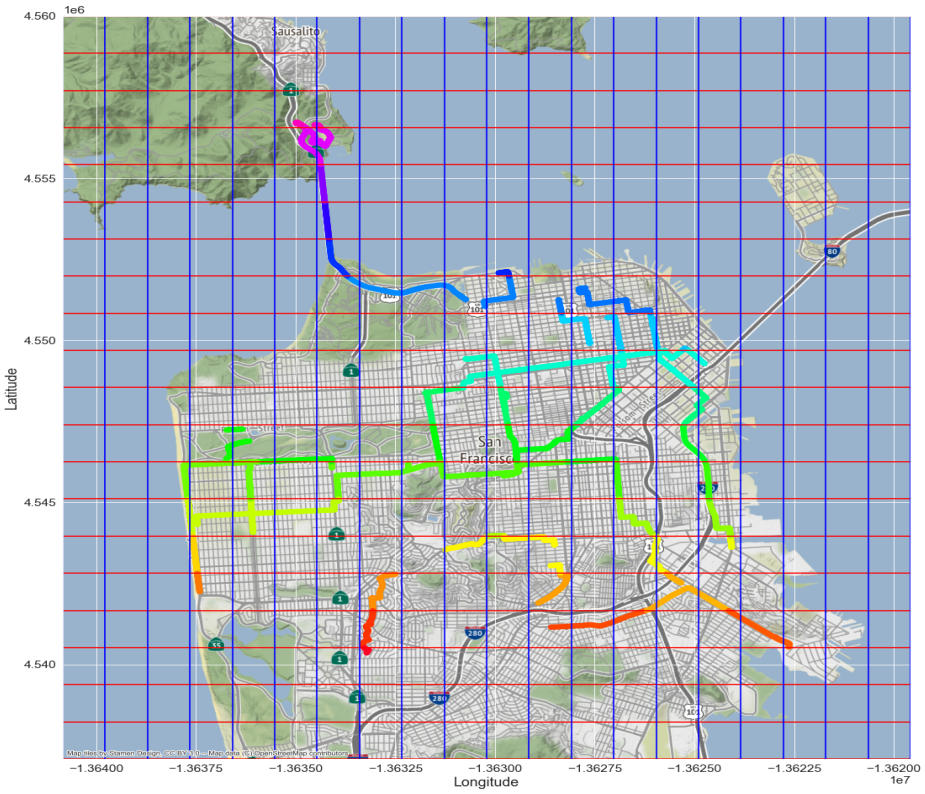

Figure 3.4: San Francisco city divided into regions (cells)

Figure 3.4 illustrate the San Francisco map divided into regions.

3.4.4.3 Preprocessing

In order to model the phenomenon described above, we must prepare the data so it will be able to be introduced into the Machine Learning model that will be described below. The input data of the model will be transformed from their original values to values which varies between 0 and 1, and this to allow the model to converge. the output values that are the cell identifier for each observation, will be prepared in another way, due to the fact that the model is a classification model. The output values will be prepared in using "One Hot encoding", which allows each class to be encoded in a column. The input data of the model will be the current and the previous values of the GPS position of the vehicle, each observation will be in the form of a matrix so that they can be processed by the layer LSTM and CNN, this matrix will have for dimension [previous values of the GPS position, variable number in inputs], the dimension of the input database is [number of observations, values previous of the GPS position, variable number in inputs]. The output data will be the classes where there is at least one single observation, the output dimension will be [number of observations, number of classes].

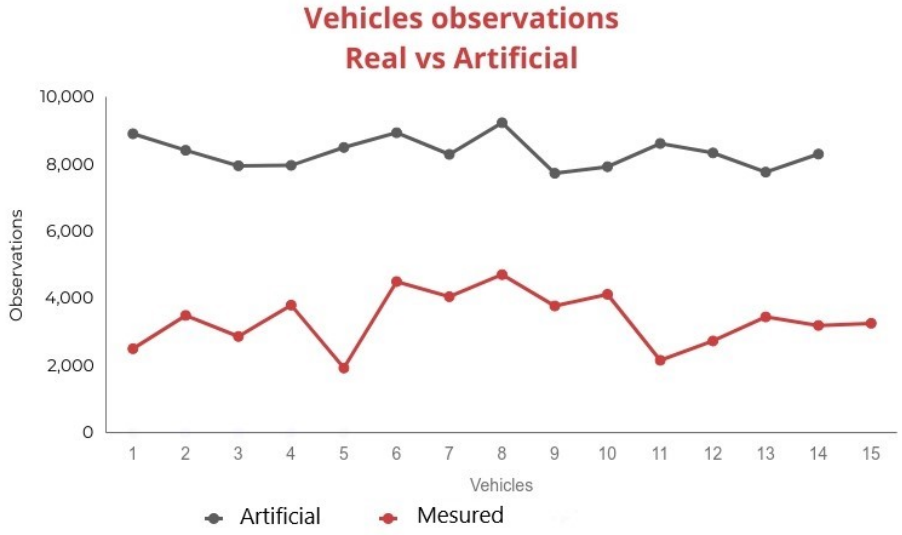

Figure 3.5: Vehicules Observations

3.4.4.4 Learning Settings

The model is trained using the "Adam" method, and using as objective function "binary cross entropy". 80% of the database will be used to calibrate the model, and 20% to test this performance, and 20% of the data training will be used for the validation of the model during its training. We used 4 metrics and 1 graphic criterion to judge the model performance, the numerical criteria are:

- Precision
- Recall
- F1-score
- Acurracy

3.4.4.5 LSTM Model

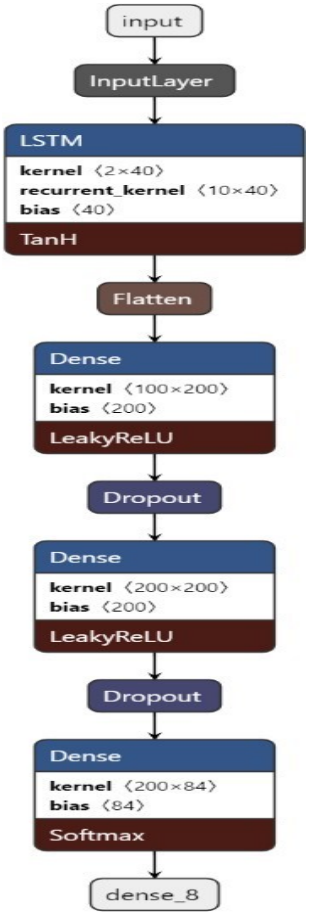

Figure 3.6: LSTM Model

Figure 3.6 illustrates the architecture deployed to generate the LSTM model.In fact after a tuning step the chosen model is composed of four layers :

- The architecture includes as a first layer an LSTM layer, in order to process temporal information and autocorrelation between observations.
- Flaten Layer reduces the LSTM dimension from a 3-dimensional vector to a 2 vector dimension so that they can be processed by layers of neural networks dense.
- The result will be transmitted to a sequences of 2 layers of dense neurons with a leaky ReLu activation function which has a parameter alpha =0.3 . The function returns 0 if it receives any negative input, but for any positive value x, it returns

that value back. Thus it gives an output that has a range from 0 to infinity, followed by a Dropout layer, which randomly turns off certain neurons in the layer previous to avoid over-learning.

- The last layer is the layer of output with 1 neuron which has the Softmax activation function which is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector.In our case the cell ID which have the highest probability will be choosen as the predicted cell or region.

3.4.4.6 CNN Model

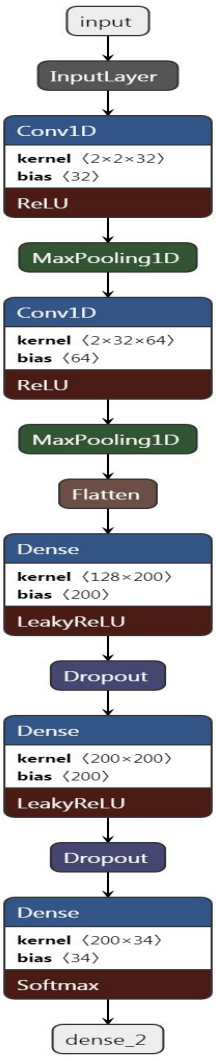

Figure 3.7: CNN Model

Concerning The CNN architecture Figure 3.7 reports its details and its compse of :

- The architecture of the CNN model includes as the first CNN layer with an equal number of filters to 32 and a "ReLu" type activation function, in order to process the temporal information of observations.
- Flaten Layer reduces the LSTM dimension from a 3-dimensional vector to a 2 vector dimension so that they can be processed by layers of neural networks dense.
- The next layer is a Maxpooling layer, in order to compress the information from the previous layer. The following layer is also composed of a CNN layer with a filter equal to 64 followed by a Maxpooling layer. The information from the Maxpooling is reduced from 3 to 2 dimensions
- The rest of the architecture is the same as for the previous, 2 dense neural layer suites with a leaky ReLu activation function which as parameter alpha = 0.3, followed by a Dropout layer, The last layer is the layer output with 1 neuron which has the Softmax activation function.

In order to calculate the output dimension of the CNN layer we use the following formula:

$$\text{output} = [(W - K + 2P)/S] + 1$$

W is the dimension of the input data
K is the dimension of the kernel
P is the dimension of the padding
S is the dimension of the stride

CHAPTER 4

RESULT AND DISCUSSION

In this chapter, we summarize some representative results of both datasets and the used models LSTM and CNN, we compare the two models on the datasets.
The experimental strategies will include respectivly:
1-Testing the performances of the Deep Learning as a method to generate forecasting models using real data and comparing LSTM vs CNN performances.
2-Comparing best models according to the nature of data (artificial, real)
according to the results achieved we will present some recommendations on the conditions under what the artificial fulfills expectations.

4.1 LSTM VS CNN

Using real data Table 4.1 and Table 4.2 reports the performances of generated models (LSTM and CNN) for the respective peroids 1,5,10,15 and 30 min .

		1 Min	5 Min	10 Min	15 Min	30 Min
Real	CNN	0.74	0.59	0.47	0.44	0.41
	LSTM	**0.75**	**0.59**	**0.49**	**0.44**	**0.4**

Table 4.1: Accuracy for LSTM and CNN for differents timestamps (15 vehicules)

The result of the modelling show that the performances is good to excellent for the predection of the position at 1 min intervals on most cases of studies .

		1 Min	5 Min	10 Min	15 Min	30 Min
Real	CNN	0.71	0.54	0.40	0.34	0.30
	LSTM	**0.73**	**0.59**	**0.46**	**0.44**	**0.4**

Table 4.2: F1 score for LSTM and CNN for differents timestamps (15 vehicules)

We also notice that the models using LSTM are a little more efficient than the models using CNN, and this is due to the fact that in order to avoid CNN over-learning, we have preferred lowered its number of iterations in the training phase. Because models using CNN are much more sensitive to overfitting than models using LSTM.

We also note that the difference between CNN layers and LSTM layers, is that CNN layers have a much more global analysis of the series of observations, and allow us to extract much more information, which leads CNNs to model global trends more effectively, by compared to LSTM layers which have an ability to process information in the short term, due to their structure. Although this rule is not true in all cases, and there are examples where the contrary is true. An in-depth analysis is therefore required to better understand the operation of the models used.

Our results confirmed the claims of W.Liu and Y.Shoji [35] [36].

4.2 Real Dataset VS Artificial Dataset

In this section we will conceder the best performances models in occurance the LSTM model with prediction periods 1min,5min,10min,15mi and 30 min

Figure 4.1 shows the compared results of the performances of LSTM models according to the real and artifical dataset in term of F1 score mesure .

more details are reported in Table 4.3 in terms of all metrics .

			1min	5min	10min	15min	30min
Precision	Artificial	**LSTM**	**0.85**	**0.79**	**0.76**	**0.76**	**0.68**
	Real	**LSTM**	**0.73**	**0.53**	**0.41**	**0.35**	**0.28**
Recall	Artificial	**LSTM**	**0.87**	**0.82**	**0.79**	**0.78**	**0.80**
	Real	**LSTM**	**0.75**	**0.59**	**0.46**	**0.44**	**0.40**
F1score	Artificial	**LSTM**	**0.85**	**0.79**	**0.76**	**0.76**	**0.66**
	Real	**LSTM**	**0.73**	**0.59**	**0.46**	**0.44**	**0.40**

Table 4.3: Weighted average for Precision,F1 score and Recall for LSTM and CNN for differents timestamps (15 vehicules)

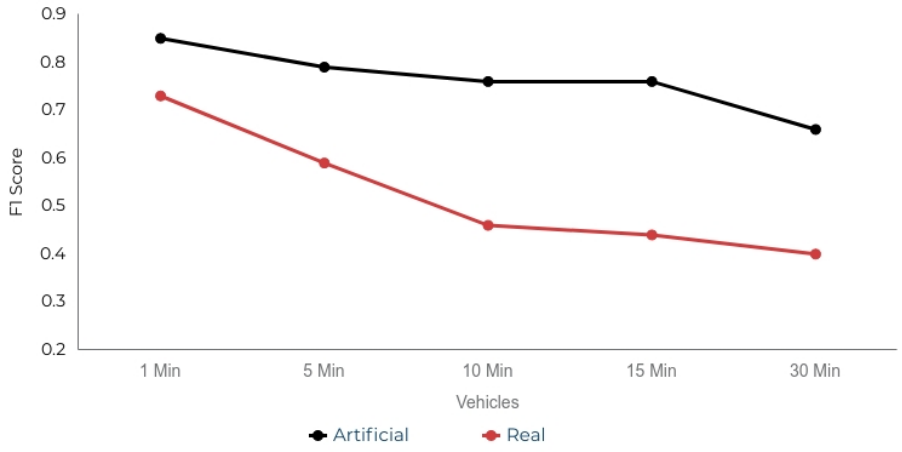

Figure 4.1: LSTM and CNN accuracy performance

The results showed that the artificial resilts are the best in the terms of all metrics in addition they are more stable when varing the time that lstm models using real data. The results of the modelling show that the performance is good to excellent for the prediction of the position at 1min intervals on most cases of studies, whether artificial or well measured and for both types of model. But the further the prediction goes in time (5min, 10min, 15min, 30min) plus performance drops drastically in the case of data measured, which is not the case for artificially generated data, due to the fact that the measured data are not distributed equally within the mesh cells ("unbalanced datasets"),

GPS data focuses on certain regions earlier than others.

One can observe that artificial data are more stable due thier balanced nature to those of real dataset ,in the next section we will try to resolve this probleme.

4.3 Unbalanced Datset Issue Resolution

As shown above measured data are not distributed equally within the mesh cells which results unbalanced dataset problem, Figure 4.2 show difference between Weighted and macro avg caused by unbalanced dataset for real data.

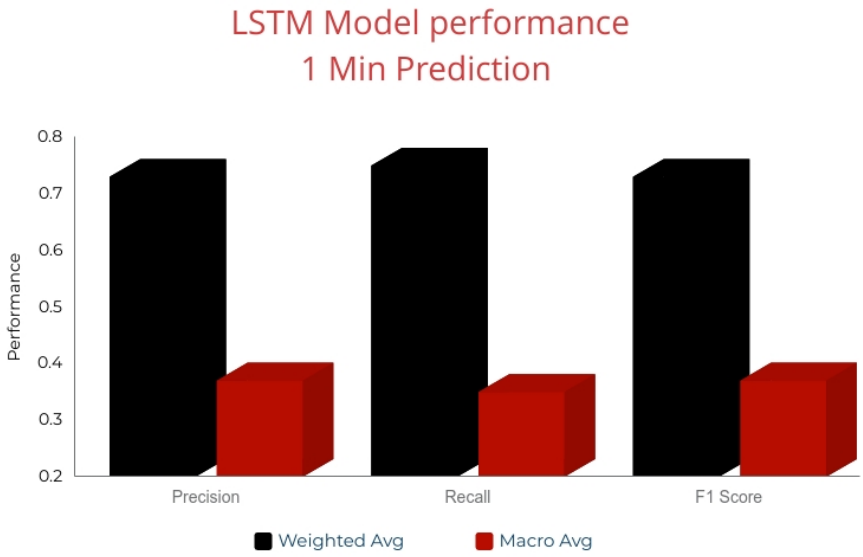

Figure 4.2: LSTM Model perfeormance 1 min prediction in Real Dataset

The results shows that there is a huge difference between weighted and macro avg due to unbalanced dataset problem, In order to balance the distribution of the data on each cell we have restricted our study to the area of downtown San Francisco where the observations are largely concentrated in this area as shown in Figure 4.3.

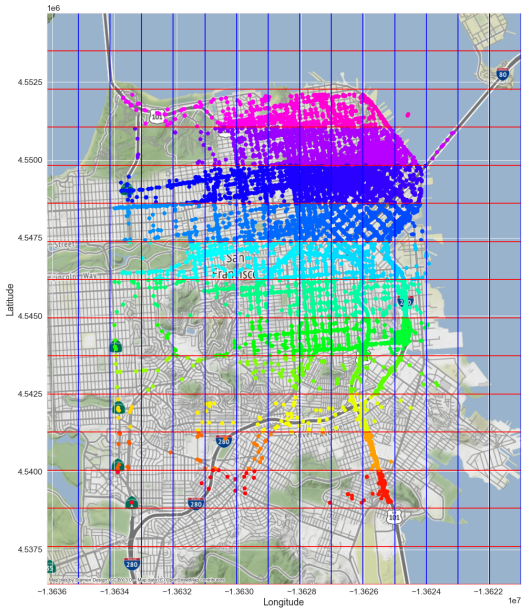

Figure 4.3: San Francisco map mesh refinement

The results shows in Figure 4.4 that the distribution of observations on cells improves much more the difference between micro and macro average it reduced then by 52% . Unfortunately performances drops drastically as the number of observations for each cell is quite low.

The performances drops by :

- Accuracy 37%
- Recall 35%
- Presecion 39%
- F1 Score 41%

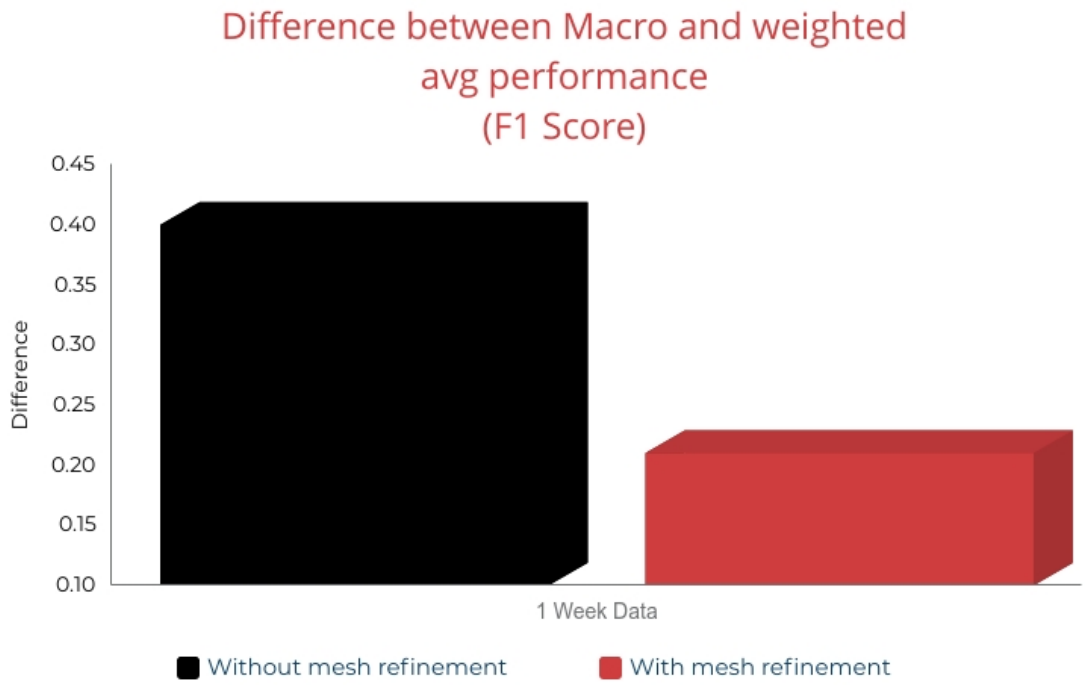

Figure 4.4: Difference betewen Macro and weighted avg performance of F1 score

In order to increase the performance of the models, we have reduced the number of cells for the area (the amplitude of the longitude and latitude being divided by 15 instead of 20). We observe that the performances increase but are not important enough. So in order to overcome this problem we used the data recorded for 1 month (instead of a week as shown in Figure 4.6). We observe that the performance has increased while keeping the same distribution (although the data remains unevenly distributed over the cells). The results as shown on Figure 4.5.

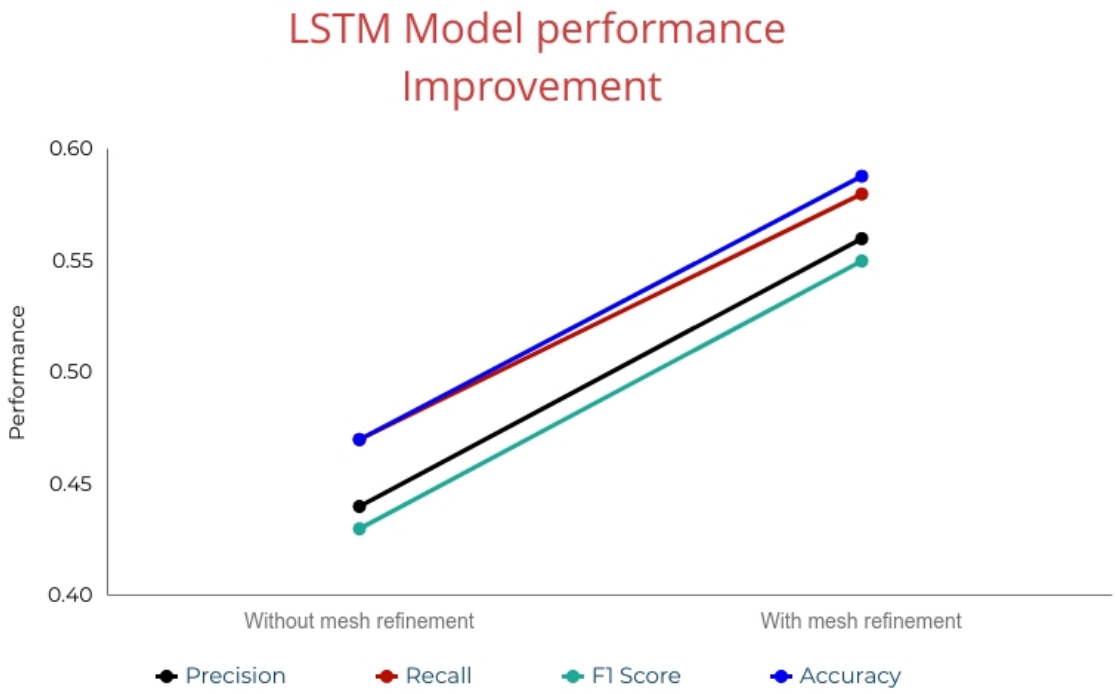

Figure 4.5: LSTM Model performance Imporvement

The performance has increased by :

- Accuracy 23%
- Recall 23%
- Presecion 27%
- F1 Score 27%

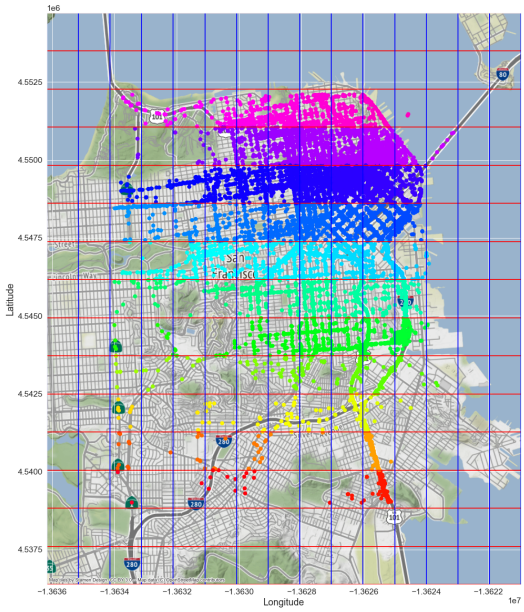

Figure 4.6: San Francisco with one month observation

The main result shown that :

-Deep Learning models gives satisafactory result in case where prediction in short-term.

-LSTM model a little more efficient than the models using CNN.

-Artificial data can be relied on to generate prediction model under some conditions.

4.4 Conclusion

In this chapter we have discussed the emprical of our proposed deep learning based forcasting approches and solutions for the occured issues.

The results show that the refinement of the mesh requires a larger number of observations to achieve satisfactory performance. which pushed us to widen the area covered by each cell of the mesh, and to increase the number of observations, which allowed us to improve the performances, compared to the previous case.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this research work we have addressed vehicle trajectory prediction in urban trafic network using Deep learning.
In fact we have designed two models (LSTM/CNN).
The two approaches are experimented on two different datasets real and artificial. We can conclude that the models used would be able to predict the position of the vehicles, and The observed data are unevenly distributed over the studied area and therefore over the cells of the mesh used.
We would also like to point out that the models trained on the artificially generated data have better performance than that of the observed data, because of unblanced dataset.
Although the proposed goals and the exposed contributions have been achieved, this project opens some further research lines to extend the obtained results.In what follow we expose some of the most interesting ones:

- The first is to create a dynamic mesh algorithm that adapts to the concentration of observations, the more the observations are concentrated in one place, the more the mesh will be fine, the more the data is dispersed the coarser the mesh be.
- The second solution consists of the generation of observations by simulation in the areas where data measured are few to counterbalance the effect of irregularities in the data.

BIBLIOGRAPHY

[1] **Rosaldo J. F. Rossetti, Ronghui Liu, and Shuming Tang.** Guest Editorial Special Issue on Artificial Transportation Systems and Simulation. IEEE Transactions on Intelligent Transportation Systems, 12(2):309–312, June 2011

[2] **G. Cookson and B. Pishue** ,2017 INRIX Global Traffic Scorecard," 2018

[3] **J. Kwon and P. Varaiya**, "Effectiveness of California's High Occupancy Vehicle (HOV) system," Transportation Research Part C: Emerging Technologies, vol. 16, no. 1, pp. 98 – 115, 2008.

[4] **Mobility and Transport**, https://ec.europa.eu/transport/themes/urban/urban_mobility_en,consulted on mai 2021

[18] **Lemagit**, https://www.lemagit.fr/conseil/Machine-Learning-comment-choisir-le-bon consulted in mai 2021

[5] **Economist**,https://www.economist.com/graphic-detail/2018/02/28/the-hidden-cost-of-congestion, consulted in mai 2021

[6] **Yann LeCun et al**. "Generalization and network design strategies". In: Connectionism in perspective 19 (1989), pp. 143–155.

[7] **Yi-Tong Zhou et al**. "Computation of optical flow using a neural network". In: IEEE International Conference on Neural Networks. Vol. 1998. 1988, pp. 71–78

[8] **David E Rumelhart et al**. "Learning representations by back-propagating errors". In: nature 323.6088 (1986), pp. 533–536

[9] **Yoshua Bengio et al**. "Learning Long-Term Dependencies with Gradient Descent is Difficult". In: IEEE Transactions on Neural Networks 5.2 (1994), pp. 157–166. ISSN: 19410093. DOI: 10.1109/72.279181.

[10] **Ilya Loshchilov et al.** "Sgdr: Stochastic gradient descent with warm restarts". In: arXiv preprint arXiv:1608.03983 (2016).

[11] **Sepp Hochreiter** et al. "Long short-term memory". In: Neural computation 9.8 (1997), pp. 1735–1780

[12] **J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi,** "DNN-based prediction model for spatio-temporal data, " Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2016, p. 92.

[13] **A. Stathopoulos, M. Karlaftis**, "Temporal and spatial variations of real-time traffic data in urban areas," Transp. Res. Rec. J. Transp. Res. Board, vol. 1768,2001, pp. 135–140.

[14] **H. van Lint, M. Schreuder M.S.c**, "Travel time prediction for variable message sign panels: Results and lessons learned from large-scale evaluation study in the Netherlands," Transportation Research Board 85th Annual Meeting, no. 06–2045, 2006.

[15] **Z. Sun, G. Bebis, R. Miller**, "Monocular precrash vehicle detection: features and classifiers," IEEE Trans. Image Process, vol.15, 2006, pp. 2019–2034

[26] **D. Acunzo, Y. Zhu, B. Xie, G. Baratoff,** "Context-adaptive approach for vehicle detection under varying lighting conditions," 2007 IEEE Intelligent Transportation Systems Conference, IEEE September 2007, pp. 654–660.

[35] **W. Liu and Y. Shoji,** "'Edge-assisted vehicle mobility prediction to support V2X communications"IEEE Trans. Veh. Technol vol. 68, no. 10, pp. 10227–10238, Oct. 2019

[36] **W. Liu and Y. Shoji,** "'DeepVM: RNN-based vehicle mobility prediction to support intelligent vehicle applications"IEEE Transactions on Industrial Informatics, 2019