Bachelor Thesis

Department of Computer and Information Science

Machine learning for early detection of traffic congestion using public transport traffic data

Author:
Berta Serracanta Pujol

Supervisors:
Mikael Asplund
Jordi Casademont Serra

June 2019

Contents

List of Figures

1 Introduction

1.1 Motivation

Nowadays, global warming is becoming one of the largest problems in the world. Since the appearance of industry, climate change effects such as a raise in temperatures, retreat of glaciers and wildfires are becoming a reality. One of the main factors that harms the environment is traffic in cities: almost 15% of the greenhouse gas emissions is produced by this sector [1]. This, not only causes harmful emissions of particles, but also the occupation of large valuable spaces for cities, among others.

In order to reduce this polluting factor, a shift in the actual mindset needs to be done. An interesting start point would be abandoning private transportation to give way to other public or non-polluting means. In favour of this change, it is necessary to improve the public transport service that is currently being offered. Better time prediction systems, a reduction in the length of journeys and more comfort for users are some examples of milestones that are required for an improved public system.

With the appearance of state-of-the-art technologies such as machine learning a lot of opportunities have opened up and some changes can now start to be implemented. Forecasting the traffic state benefits not only citizens but also companies. Better planned cities or dynamic public transport lines are some examples that would lead to a smarter and greener city. The desire of making this predictions is starting to become a reality with the appearance of machine learning algorithms.

1.2 Aim

The aim of the thesis is not only to provide a machine learning based solution for early detection of traffic jams so that in the future migration strategies could be employed but also analyse and comprehend the potential of the data provided by local Östgötatrafiken buses for the traffic detection. In other words, the main goal is to propose, implement, test and optimise a machine learning algorithm based on data collected from regional buses that is able to perform predictions on the future state of the urban traffic.

1.3 Research questions

In order to put into words the main goal of the project, forecasting the traffic state, two research questions were posed. Since the project had two big parts it was clear that there had to be one for each part. The first one was about the data and with the intention to determine if the collected data was fit for the problem. The second question referred more to the machine learning part, meaning to question if this type of networks were able to learn the time dependencies that represented the data collected.

1. Is public transport real-time speed data a good representative of the traffic state? Does this data behave as a periodic function?
2. Can a machine learning algorithm predict the future speed based on previous measurements?

1.4 Approach

The approach taken for solving the questions in the section above is developing a neural network based architecture that can perform as a model to this particular scenario.

There are two main problems that need to be solved. First the implementation of a machine learning architecture capable of performing predictions based on real-time GTFS data [2]. The second is the optimisation of such algorithm to achieve the maximum accuracy. The aim is once the framework has been implemented is to gradually analyse and adjust the network's hyper-parameters and chose the ones that provide the best match.

The planned procedure for the thesis is as follows:

Starting on the existing related fields, previous works and procedures, an extensive research will be performed. This process includes the data acquiescence as well as the implementation of the needed metrics to adapt it for our project. Also, a broad research will be conducted on machine learning methods to optimise the approach required. Following, the test evaluation method will be discussed and critically reviewed to fit with our scenario. Furthermore, in an iterative process, the methods and algorithms for developing the neural network architecture will be studied, implemented and tested. Finally, the iterative process will finish with an evaluation and optimisation of the crucial parameters and with a final system review. At the end, possible further developments and studies will be mentioned.

1.5 Delimitations

This ten-week project does not aim to reach the definitive solution to traffic congestion predictions. However, it does intend to provide a first approach to a potential solution from which others can continue working.

The forecasts made in this project are only based on speed measurements provided by the local transport system and does not contemplate other unexpected factors such as road works, accidents or traffic restrictions. Some ideas are given to implement some upgrades to the solution proposed.

When talking about machine learning, the field is vast and in this project only a small part of it has been contemplated. More in concrete this project focuses only in the points listed below:

- Supervised learning
- Regression analysis
- Time series predictions with artificial neural networks

2 | Background

2.1 Intelligent Transport Systems

Intelligent transport systems (ITS) are applications of sensing, analysis, control and communication engineering in road transportation in order to improve safety, mobility and efficiency. With the aim to provide innovative services, ITS are applied in many areas such as infrastructure, users, traffic and mobility management, interfaces with other modes of transport and vehicle communications. In other words, is the application of technology to the movement of goods and people.

It is an expanding international market, since eventually many driving functionalities are going to be automated. Consequently, features such as achieving the maximum efficiency or reducing traffic incidents are to become fundamental.

ITS vary in terms of the technologies applied. From basic systems such as car navigation, traffic signal control systems, speed cameras and automatic number plate recognition to more complex applications like road states (deicing systems), assisted parking guidance and many others.

They also include a wide range of applications that process and share information to ease congestion, improve traffic management, minimise environmental impact and increase the benefits of transportation to commercial users and the public in general.

There are five main areas of applications. The first type of application is the automation of road enforcement by using cameras, radars, LIDARs and sensors that control driver infractions committed while travelling. The clearest example is the speed limit enforcement integrated with the number plate recognition system. Linked with this application there is the variable speed limit control, which dynamically changes the road speed regulation depending on the actual state of the traffic. There is also the management of emergency vehicle systems which tries to optimise the response since the occurrence of an incident until its resolution. Dynamic traffic light sequence and collision avoidance systems are the last main areas but there are others too numerous to mention. Together they are helping smart mobility models to emerge.

Traffic state prediction by analysing vehicles speed is a critically important topic in terms of Intelligent Transport Systems since it is the base of many other applications where knowing the road state is essential. In other words, it is essential for bringing smart cities to live.

2.2 Machine learning

Machine learning is an application from artificial intelligence dedicated to design, analyse and develop mathematical algorithms and techniques that are capable of making decisions from sets of data (training data).

Any system that is considered intelligent must have the ability to learn, that is, to automatically improve with the experience. The programs used are learning systems capable of acquiring high level knowledge and strategies for problem solving through examples, in a way similar to the human mind. Tom Mitchell, computer scientist known for his contributions to the advancement of machine learning, said: "A computer program is said to learn from experience (E) with respect to some class of tasks (T) and performance measure (P), if its performance at tasks in (T), as measured by (P), improves with experience (E)"[3].

LSTM networks can learn the features of time series but it is not an easy task, particularly when there is a short number training values.

2.2.1 Learning methods

Machine learning algorithms are classified according to what the program is expected to learn and also according to the degree of interaction with the user. There are three main styles: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning.

Supervised learning

This type of learning requires a supervisor. The input data used by the model needs to be labelled with the expected output. These labels for output vector are provided by the supervisor. Often, these supervisors are humans, but machines can also be used for such labelling[4]. Through iterative optimisation, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs unseen by the model.

Supervised learning problems are categorised into regression and classification problems. In a regression problem, the objective is to predict results within a continuous output, mapping input variables to some continuous function. In a classification problem, the results predicted consist in a discrete output. In other words, mapping input variables into discrete categories. This style of learning provides error measurements. After the prediction has been made, feedback is collected and and the error is computed.

Below, some examples of supervised learning algorithms:

- Linear regression
- Logistic regression
- Naive Bayes
- Neural Networks
- Similarity learning
- Support Vector Machines

Unsupervised learning

In unsupervised learning, we lack supervisors[4]. In contrast with the learning method above, unsupervised learning is provided with a set of training examples that have not been classified nor labelled. In this case, the algorithm must be able to find similar patterns and structures in the training data in order to be able to label, group or classify it. Instead of responding to feedback, these type of algorithms identify the similarities in the data and react based on the presence or absence of such commonalities in each new piece of data.

Some of the more common used algorithms are the following:

- Clustering
- Anomaly detection
- Neural Networks
- Latent variable models

Semi-supervised learning

Semi-supervised learning algorithms combine the two types of previous algorithms in order to generate a suitable function or classifier [4]. A combination of marked and mostly unmarked data is used as a training set. This type of algorithms usually increases the accuracy of unsupervised learning but without the time and costs needed for supervised learning.

Reinforcement learning

The reinforcement learning method aims at using observations gathered from the interaction with the environment to take actions that would maximise the reward or minimise the risk[4]. Once the action taken by the system gets feedback from the surroundings (a gratification or a penalty depending on whether the action has been successful or not). That is, input information is the result you get from outside as a response to your actions. It is learned through trial-error and, due to this, a high number of repetitions is required. Reinforcement learning is mostly used in non-deterministic and changing environments where good error measurements are difficult or impossible to perform.

Reinforcement learning algorithms include:

- Monte Carlo
- SARSA (State–action–reward–state–action)
- Q-learning
- DQN (Deep Q Network)

2.3 Artificial neural networks

Artificial neural networks, usually noted as ANN, are a paradigm of learning and automatic processing inspired by the way the nervous system of animals works. To mimic that behaviour, an ANN consists in different nodes (artificial neurons) interconnected to transmit signals from one to

another. The main objective is to get the machines to give answers that are similar to those that are capable of giving the brain.

As mentioned before, the network consists of connections and everyone of them is assigned a weight. Each neuron receives a series of inputs and gives and output determined by three main functions: a input function, an activation function and a transfer function. The objective is to successfully tune all the different parameters to achieve the desired output.

The neurons are organised in layers, as depicted in figure 2.1. The input layer (Green) is the one that gives the inputs to the network. Hidden layers (Orange)can take inputs (from the input layer or from other hidden layers) and multiply its weights by the inputs received. That result will be passed to the next hidden layer or to the output layer if it is the case. The output layer (Blue) takes its inputs from the last hidden layer and computes the final network output.

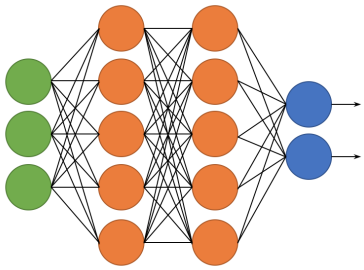


Figure 2.1: Neural network architecture with two hidden layers

There are three main features to be considered when developing a neural network architecture: the number of neurons in the input layer, the number of hidden layer and its neurons and the neurons of output layer. The input layer size will be determined by training samples on the database such as the pixels value of an image or the words of a sentence. Hidden layers do not have a determined size, it has to be fixed depending on the accuracy of the performance of the network, by adjusting in every run. Finally, the output layer size will have the same length as our desired output.

Another fundamental aspect when developing a neural network is the training process. After having set the architecture and having the outputs of the network for a given input, a need for an algorithm that evaluates how close the value is to the ground truth appears. This information helps with adapting the weights set on values to get more accurate results for the following iterations. The way to conduct this problem in machine learning is using loss functions. The loss function measures the quality of a particular set of parameters based on how well the output of the network agrees with the ground truth labels in the training data.The loss function does not want to measure the entire performance of the network against a validation/test dataset. It is used to guide the training process in order to find a set of parameters that reduce the value of the loss function. With this appears the training process, which consists in finding a set of parameters which make the loss as small as possible and then changing parameters at a rate determined by the partial derivatives of the loss function. The algorithm in charge of this steps is the stochastic gradient descent, which ideally modifies the parameters in small steps until reaching the minimum during backpropagation passes.

Depending on how the connections between the neurons are made there are different types

of networks. The first ones that were developed are called feed forward networks and follow a similar structure as figure **??**. Its connections between nodes do not form a cycle, meaning that the information only travels in one direction: forward. In contrast, recurrent neural networks (RNN) have connections that form loops which allow temporal dynamic behaviour. This type of networks have an internal state also referred as memory and are capable of solving more complex problems such as handwriting recognition. In other words, sometimes they are able to connect previous information to the present task.

2.3.1 Long Short-Term Memory networks

Unfortunately, RNN are not capable of making long term relations between the information due to the gradient descent algorithm used as a learning algorithm when training the network. When implementing this algorithm, which is used for the optimisation and minimisation of the cost function of the whole system, the network faces a problem known as vanishing gradient. The gradient descent becomes increasingly inefficient when the temporal span of the dependencies increases [5].

Long Short-Term Memory networks, also known as LSTM, are a kind of RNN which are capable of performing this ling-term dependencies between the data. Introduced in 1997 by Hochreiter & Schmidhuber [6], LSTM networks are well-suited to making predictions based on time series data since they were explicitly designed to avoid the vanishing gradient problem that faced traditional RNNs.

All recurrent neural networks have the form of a chain of repeating modules of a neural network. In standard RNNs they have a simple structure consisting of a single layer. Although LSTMs do not vary the chain structure, the repeating module has a completely different structure: it consists in four layers that interact between them.

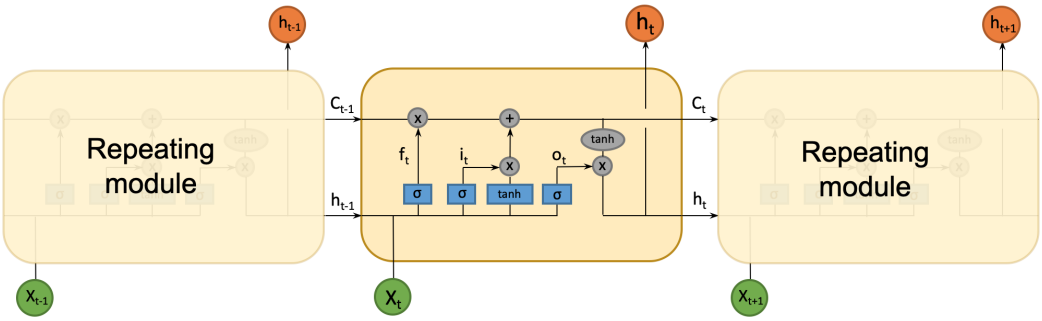


Figure 2.2: LSTM chain structure

In figure 2.2, each arrow carries a vector from the output of one cell (repeating module) to the input of the next one. Each blue rectangle represents each neural network layer and the grey circles represent point-wise operations.

The main idea of LSTMs is the cell state, represented by the top horizontal line. The information

flows through the entire chain while some linear interactions are performed. These interactions are called gates and consist in a sigmoid layer and a point-wise operation. The sigmoid layer outputs a value between zero and one which describes how much of every component should go through. This cell state is computed as follows:

$$
\begin{aligned}
f_t &= \sigma\big(x_t U^f + h_{t-1} W^f\big) \\
i_t &= \sigma\big(x_t U^i + h_{t-1} W^i\big) \\
o_t &= \sigma\big(x_t U^o + h_{t-1} W^o\big) \\
\tilde{C}_t &= \tanh\big(x_t U^g + h_{t-1} W^g\big) \\
C_t &= \sigma\big(f_t * C_{t-1} + i_t * \tilde{C}_t\big) \\
h_t &= \tanh(C_t) * o_t
\end{aligned}
$$

The first gate, called forget gate (f_t), decides what information is no longer required from the previous the cell state. Matrices W and U contain, respectively, the weights of the input and recurrent connections. The second regulator, the input gate layer (i_t), decides which values are going to be updated and is formed by a sigmoid function and after combined with a tanh layer that gives potential values that could be added to the state ($\tilde{C}_t$). After these two gates the cell state is updated (C_t), as depicted in the left half of the repeating module of figure **??**. Finally, the output gate (o_t) controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit.

2.4 Measures of performance (MAE, MAPE, RMSE)

There are multiple ways of analysing the performance of a neural network implementation. Time and accuracy are the main aspects when determining if an application is good enough. The time constraint can be seen as how long it takes to train a model or the time it takes to answer a query and so on. The accuracy of a model however, is a measure that determines how good the predictions performed by a model are.

When choosing an accuracy metric the type of learning needs to be taken into account. Since time series prediction is a supervised regression learning problem, the metrics need to be for continuous variables.

The main four regression metrics are the following:

Root Mean Squared Error (RMSE)

The root mean squared error represents the sample standard deviation of the differences between predicted values and observed values (called residuals). RMSE is always non-negative, and a value of 0 (almost never achieved in practice) would indicate a perfect fit to the data. In general, a lower RMSE is better than a higher one. The effect of each error is proportional to the size of the square

error meaning that major errors have a larger effect on the RMSE. However, even after being more complex and biased towards higher deviation, RMSE is still the default metric of many models because loss function defined in terms of RMSE is smoothly differentiable and makes it easier to perform mathematical operations. Mathematically, it is calculated using this formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Mean Absolute Error (MAE)

MAE is the average of the absolute difference between the predicted values and observed value. The MAE is a linear score which means that all the individual differences are weighted equally in the average. Its range varies from zero to infinity, being zero the ideal value. It is easier to understand and interpret MAE because it directly takes the average of offsets whereas RMSE penalises the higher difference more than MAE. The Mean Absolute Error is given by:

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

Confidence interval

The confidence interval is a range between two values around a sample parameter in which, with a given probability (or level of confidence), that parameter will be placed in the population. In other words, a range of values so defined that there is a specified probability that the value of a parameter lies within it. A sample parameter that is usually determined by the confidence interval is the average. The desired level of trust is established by the researcher, not determined by the data properties. More commonly, the confidence level of 95% is used. However, other levels of confidence can be used, for example, 90% and 99%. At a lower level of confidence the interval will be more accurate, but a greater error will be made. For determining this interval, the data is assumed normally-distributed and in our case with known mean and standard deviation.

C	z^*
99%	2.576
95%	1.96
90%	1.645

$$(\bar{x} - z^*\frac{\sigma}{\sqrt{n}}, \bar{x} + z^*\frac{\sigma}{\sqrt{n}})$$

Table 2.1: Confidence interval formula

2.5 Time series data

A time series is a sequence of data points, observations or values measured at certain times and ordered accordingly. Usually, but not always, these points are equally spaced in time. Thus it is a series of discrete data and commonly used in a large range of fields such as statistics, mathematical finance, signal processing or weather forecasting.

The analysis of this type of data includes methods that help extracting representative information regarding its origins, relations or forecasting its behaviour. Time series analysis accounts for the fact that data points taken over time may have an internal structure (such as autocorrelation, trend or seasonal variation) that should be accounted for. These methods can be divided in two classes: frequency-domain and time-domain.

Time series forecasting is an important area of machine learning since time plays a role in machine learning datasets. When treating a dataset there can be different goals: understanding it or making predictions. The first one can help achieving the second one but it is not always required."In descriptive modeling, or time series analysis, a time series is modeled to determine its components in terms of seasonal patterns, trends, relation to external factors, and the like. ... In contrast, time series forecasting uses the information in a time series (perhaps with additional information) to forecast future values of that series." [7] The skill of a time series forecasting model is determined by its performance at predicting the future.

2.5.1 Neural networks for forecasting

Neural networks are a popular framework when facing supervised learning problems, since a system of weighted additions and differentiable functions can learn complex relations. Time series predictions are just one application of where neural networks are currently being used. The pioneer work of developing and applying backpropagation (chain rule) to forecasting was made by Werbos [8] which outperformed traditional methods when analysing to short memory series and similar results for long memory.

LSTMs, although not being its primary application, are often used for time series forecasting [9]. Some of the most remarked works are using peephole connections, a variation of LSTMs, for learning temporal distances [10]; stacking LSTM networks to detect anomalies in time series [11] or modelling periods and missing values in time series [12].

2.6 Related Work (previous ML solutions for traffic prediction)

Zhao et al. [2016] [13] published a paper presenting an algorithm based in machine learning techniques for detecting the freeway traffic state. They propose two speed difference parameters to study the speed difference and a detection algorithm that makes real-time statements of the traffic situation. Such algorithm is tested in non-current and current congestion simulated environments

and the problem is approached as a classification between different traffic states.

An error-feedback recurrent convolutional neural network is proposed as a solution to traffic speed prediction and congestion source exploration by Wang et al. [2016] [14]. In this article, a spatio-temporal traffic speeds of contagious road segments are the input to a pre-trained network as they try to predict abrupt changes in the traffic state in Beijin. They results are compared with more traditional methods such as Auto Regression Integrated Moving Average (ARIMA) or Support Vector Machines (SVR).

3 | Method

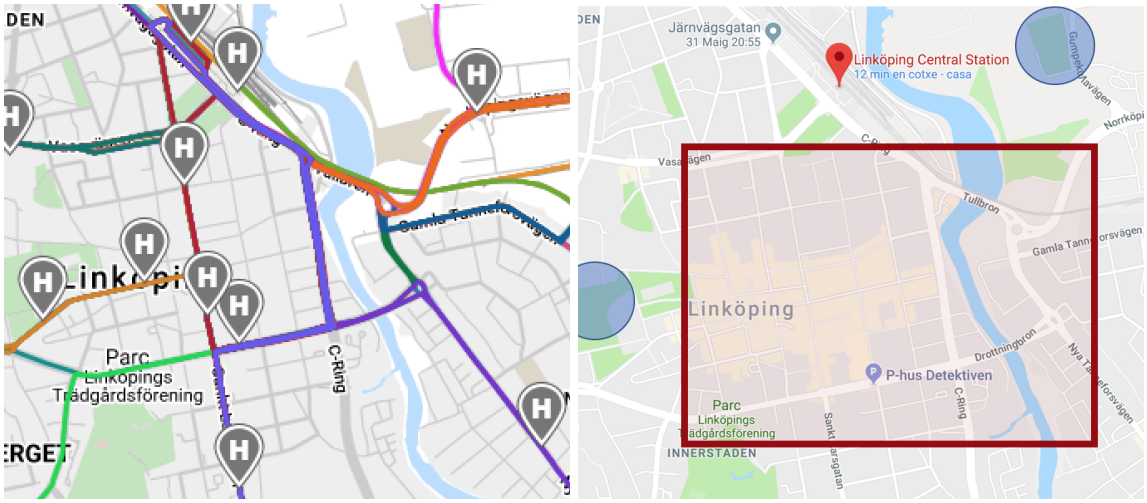
In this chapter we present the method used in the experiment that was performed. The goal of the project was to predict the future traffic state using public data information. Before anything, the project scenario definition is depicted. To achieve the project goals, a LSTM network has been implemented, tested and later evaluated. But in order to achieve that the data needs to be pre-processed to fit the network correctly which implies understanding it. Firstly, the studied scenario will be defined. Then the data characteristics and its processing will be detailed. Next, a thorough description of the experiment will be performed, which will contain details of the scenario chosen and the architecture implemented. Finally, the evaluation method used is going to be explained.

3.1 Scenario

In order to focus the project in specific topics such as the performance of the neural network and the data analysis a scenario must be define to avoid too broad or too little information.

The bus data provided by Östgötatrafiken belongs to the area of Linköping, where the company provides service. When defining the scenario, different aspects must be taken into account. First of all, it can not be a very large area since little population density does not provide enough traffic situations to be analysed. It is also necessary to consider the flow of vehicles, since a scenario located on the outskirts of Linköping would not provide sufficient data to form a dataset nor would it be representative. For these reasons, three main aspects have been taken into account when defining the stage. Firstly, the figure 3.1a showing the different bus lines offered by Östgötatrafiken since the amount of data collected was an important factor. Secondly, the position of the bus station indicated in the figure 3.1b in red, since it is the place with more frequency of lines so that it raises the number of samples. It should be considered, however, that the station can not be included in the scenario since buses transmit zero speeds in that area and would deflect real measures. And finally the location of areas with large numbers of people, marked in blue in figure 3.1b, such as SAAB Arena and Linköping Konsert & Kongress.

The final scenario latitude and longitude corner points are: [58.41525, 15.64384], [58.41525, 15.61902], [58.40224, 15.64384], [58.40224, 15.61902]. The total area consists on a square surface of 1,45km on the side resulting on a total surface of $A = 2,10km^2$



(a) Bus lines service by Östgötatrafiken (b) Final scenario (green) with important locations

Figure 3.1: Scenario definition

3.2 Data mining

In this section three main aspects are going to be illustrated. Firstly, the process of data acquiescence will be detailed. Secondly, the data processing for building a time series. Finally, the process of transforming this time series to adapt it to the neural network.

3.2.1 GTFS Real-time Data

GTFS stands for General Transit Feed Specification and is a standard that was developed by Google. This standard defines a common format for public transportation schedules and associated geographic information. GTFS feeds allow the flow of transit data between transit agencies and developers. The feeds are represented in a series of text files that are compressed into a ZIP file, and include information such as fixed-route schedules, routes, bus stop data among many other. GTFS datasets are used in a variety of types of applications, including trip planners such as Google Maps, timetable generation software, mobile applications, tools for transit planning and operations analysis.

GTFS Real-time is an extension to GTFS. This type of feed lets transit agencies provide consumers with live information about disruptions to their service (stations closed, lines not operating, important delays, etc.) location of their vehicles, and expected arrival times.

A feed may, although not required to, combine entities of different types. Feeds are served via HTTP and updated frequently. The file itself is a regular binary file, so any type of web server can host and serve the file (other transfer protocols might be used as well). Alternatively, web application servers could also be used which as a response to a valid HTTP GET request will return

the feed. There are no constraints on how frequently nor on the exact method of how the feed should be updated or retrieved. Because GTFS Real-time allows presenting the actual status of a fleet, the feed needs to be updated regularly - preferably whenever new data comes in from the Automatic Vehicle Location system.

The specification currently supports the following types of information:

- Trip updates
- Service alerts
- Vehicle positions

In this project, the focus has been on the vehicle positioning information.

Vehicle position is used to provide automatically generated information on the location of a vehicle, such as from a GPS device on board. A single vehicle position should be provided for every vehicle that is capable of providing it.

The structure of a feed message is simple: a header followed by entities. The data provided by Östgötatrafiken does not contain all the fields specified by the GTFS real-time standard, since most of them are optional and not required. The header contains, at least, the GTFS version used, a timestamp and the size of the information that carries. Each of the following entities can hold the three items mentioned above, and Östgötatrafiken provides only the latest. Vehicle position contains fields which provide information about the trip, the position of the vehicle, a timestamp and an id.

3.2.2 Processing feeds

Although all feeds are served through HTTP protocol, the data exchange format in GTFS Real-time is based on Protocol Buffers. Protocol buffers are a language- and platform-neutral mechanism for serialising structured data (similar to XML, but smaller, faster, and simpler). The data structure is defined in a gtfs-realtime.proto file, which then is used to generate source code to easily read and write your structured data from and to a variety of data streams, using a variety of languages – e.g. Java, C++ or Python.

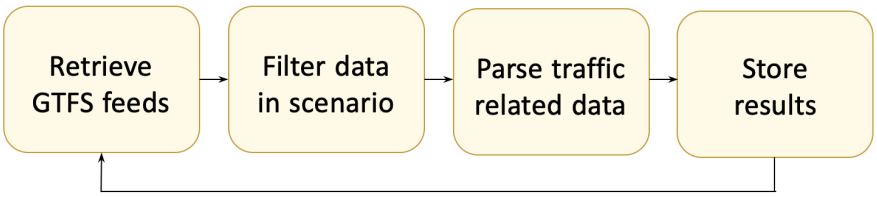


Figure 3.2: Feed processing diagram

In order to filter only the important information, a program that follows the steps mentioned in figure 3.2 has been developed.

To periodically obtain the relevant data from an instant, the program must run on a loop. This loop, which then calls the methods for parsing the data, is called every five seconds. This time value has been chosen in relation with the definition of the scenario, which will be detailed in the following section. For parsing every data feed, a call retrieving the API URL needs to be made. In order to acquire the data an API key is needed, this is provided by Trafiklab [15], a community for open traffic where all the transport API from Sweden are collected. The response is then parsed using protocol buffers and iterated to save the data required for the experiment.

Not all the downloaded feed is stored, instead it is filtered using the following criteria. Firstly, only the vehicles inside the area considered at the scenario are evaluated. Then, the interesting features related to each vehicle are extracted: latitude, longitude, vehicle id, speed and time when the measurement has been performed. These features are then saved as a line inside a comma-separated value file. Only those fields related with the traffic have been saved. Others like the bearing of the vehicle, the id of each entity (not the vehicle) or information about the trip that was being performed are not considered relevant for this problem.

The data needs to be collected for a long period of time in order to have enough training examples for the machine learning algorithm to find structure within it. After the gathering the data, some processing needs to be done in order to feed it to the neural network. This processing has been done with the Pandas Data Analysis Library: an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language [16].

3.2.3 Building a time series

Once all the data is collected, a time series needs to be build. Since the data has been gathered following a time line there is no need to order the samples. However, an important characteristic of time series is that the samples are equally spaced. A real-time feed has been saved every five seconds which in average contains ten speed samples of different vehicles. This instantaneous speed is not a suited representative of the actual traffic state, since it could have a null value due to a bus stop or traffic light. Also, there isn't available data for every hour of the day, since it is limited to the public transport schedules.

For all the reasons mentioned above, the data needs to be treated again to fit the project needs. The first thing needed is to group the data regarding different labels: date, weekday, hour, minute and speed. Then, using MATLAB software, stored in a table with the objective of reaching an array formed by the average speed of each hour ordered by time. Since the instantaneous data is not reliable when making predictions, the different samples are grouped by hour and date and then averaged. As studied in [17], performing a weighted arithmetic mean in the samples samples that only differ in the minute feature gives an accurate representation of the traffic state. The weighted arithmetic mean is similar to an ordinary arithmetic mean (the most common type of average), except that instead of each of the data points contributing equally to the final average, some data points contribute more than others. In this case this helps accounting for the difference in number of samples in each minute.

3.2.4 Time series to supervised learning data

In order to input the data collected to train or test the LSTM network, the dataset must follow some rules. First of all, since a supervised learning algorithm is being implemented, the data needs to be labelled. That means going from a list of numbers to a list of input and output patterns. Each training example of the dataset needs to follow the structure below:

$$X_{t-n_{ts}}, X_{t-(n_{ts}-1)}, ..., X_{t-n_1}, X_t, Y_{t+m_1}, ..., Y_{t+m_{ft}-1}, Y_{t+m_{ft}}$$

Being X the input data, Y the output ground truth or labels, $n = lag \cdot (1, 2, ..., ts)$ and $m = lag \cdot (1, 2, ..., ft)$. Each sample X represents a speed value in a specific time point. The sequence of inputs represents past observations from which the network will try to learn a connection to make the predictions Y. This data must be equally spaced in time and in order to define this space the parameter n is defined. The time-steps, ts, are the number of samples that will receive the network and lag is the time value difference between them. Similarly, the output features or future times, ft, are the number of values that the network will predict. A simple example, if we have a dataset of speed every hour (50 samples total) and the last six hours need to be considered for predicting the next three we will have a matrix with $ts + ft$ columns and each row will consist in $X_{t-6}, X_{t-5}, ..., X_{t-1}, X_t, Y_{t+1}, Y_{t+2}, Y_{t+3}$. It has to be taken into account that even though the original dataset had 50 training examples the final matrix will not have the same number of rows since there will be some NaN values that need to be dropped.

3.3 Network architecture

Before deciding to implement a Long Short-Term memory Network, other models used for forecasting were thoroughly evaluated. In both subsections below, the other type of architectures studied are detailed.

3.3.1 Feed Forward Networks

The first algorithm studied were Feed Forward Networks (FFN) which were the first ever developed. In this particular network the information always flows the same way, forward. It has a clear structure and it is divided by layers. As mentioned in the Background chapter, it starts at the input layer, depicted in green in figure 2.1, and the information is passed through the next layers with a specific weight for each link. The sum of the products of the weights and the inputs is calculated in each node, and if the value is above some threshold (typically 0) the neuron fires and takes the activated value (typically 1); otherwise it takes the deactivated value (typically -1). After going through hidden layers the values reach the output layer were the result is obtained. The connection of the neurons between layers never make cycles or loops. To adjust weights properly, one applies a general method for non-linear optimisation that is called gradient descent.

3.3.2 Convolutional Neural Networks

Another of the algorithms studied and considered as a possible solution for the problem presented were Convolutional Neural Networks. These type of networks share the same general structure as a FFN with input, hidden and output layers. However they are more complex since instead of just weighting and adding values more complicated operations are performed. An example of those is detailed in figure 3.3 below. They are commonly used for processing single data points such as images since they can extract features of the input matrices. This type of network is capable of finding more difficult relations between the data provided but it requires large amounts of input data.

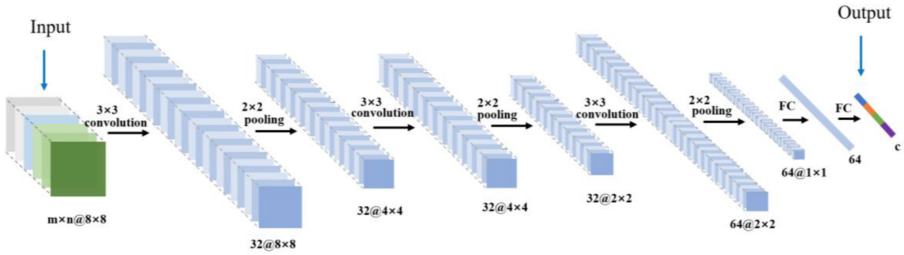


Figure 3.3: Example of a CNN architecture

3.4 Performance evaluation

In order to have a numerical method to determine the performance of the developed model, a baseline was required.

To determine the quality of the forecasts performed the reference model selected was a arithmetic mean computed using all the collected speed samples. This average, computed for each hour of the day, consisted in a 126 sample sequence that represented the 18 samples for each day of the week. For simplicity, no weekly variations were considered, meaning that four of this sequences concatenated one after the other represented a month.

This baseline serves as a comparator to the predictions made by the machine learning algorithm. In order to be able to determine if the outputed results are better or worse than the baseline, both the Root Mean Squared Error and the Absolute Error have to be computed and compared. If these errors are lower for the developed implementation then the evaluation is successful.

4 Results

In this chapter the results of the project are presented. As before, it is mainly divided in three sections consisting in the scenario, the collected data, the choice of architecture implementation and finally its evaluation.

4.1 Scenario

In order to visualise if the acquired data was relevant for the problem the coordinates that reported low speed values (lower than 3.5 meters per second) were plotted in a heatmap using the Maps Javascript API from Google. As it can be seen in figure 4.1, there are enough low-speed values to train a small neural network. From this picture, we can also see that this samples are mainly concentrated in Sankt Larsgatan.

Figure 4.1: Heatmap of high traffic areas inside the scenario

4.2 Data mining

For this project, samples of real-time traffic feeds have been taken every five seconds and each measurement consisted, on average, in 10 speed samples due to scenario limitations. This resulted in having 120 samples every minute, which turned to 7,200 samples per hour. It has to be considered that Östgötatrafiken does not provide bus service between 00:30 and 04:30 during weekdays, so 144,000 samples were gathered each day. In total, 3.423.691 speed measurements were gathered in a period of six weeks. After applying the previously described processing and computing only hours between 6:00 AM and 11:00 PM (a total of 18 hours per day), the dataset consisted in 756 speed samples chronologically ordered.

In figure 4.2 the final time series obtained is depicted:

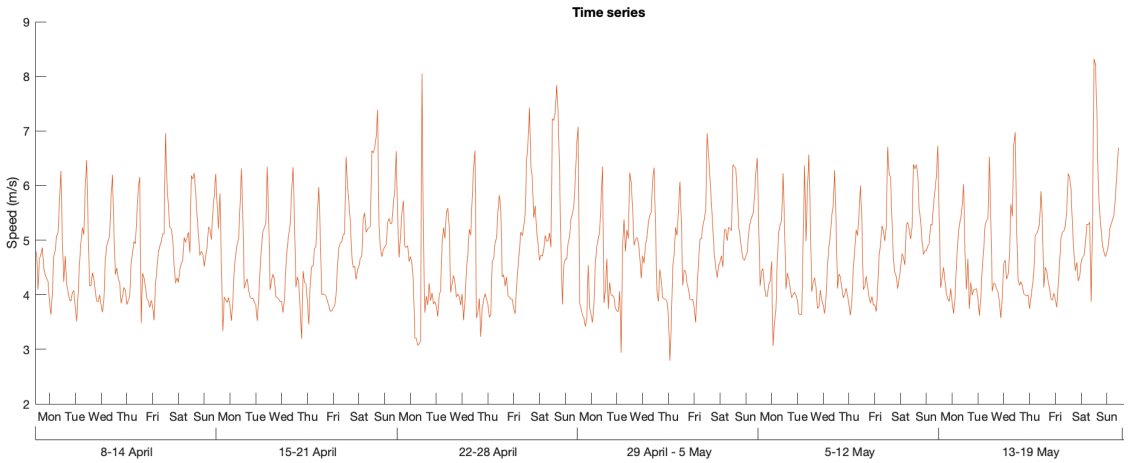


Figure 4.2: Speed time series dataset

In order to understand better the data being used as the network input, different analyses have been performed. The first thing studied was the variation of the speed during a day, depicting the results in the superior plot in figure 4.3 below. In this figure we can see the main differences between weekdays and weekends. The superior plot shows the changes in the average speed during the course of the day. It can be seen that Saturdays and Sundays report higher speed values while values from Monday to Friday are lower. For the later, the local minimum peaks are located at 7:00 AM and 4:00 PM. To numerically have an indicator of the differences between the types of days, the inferior plot in figure 4.3 has also been added. This graph displays the variance when comparing four different values. The first one is the variance between the first five days of the week. The second line depicts the variation between both days of the weekend. It can be noted that both of these lines represent the lowest variations during the day. For the remaining two lines, the comparison is between the weekend days. It can ve observed that both share the lowest values for the majority of the hours. The last two lines represent the study of speed variations of Saturday and Sunday when compared with other weekdays respectively. These approximately share the same time evolution, reaching its maximum at 7:00 AM with another local maximum at 4:00 PM.

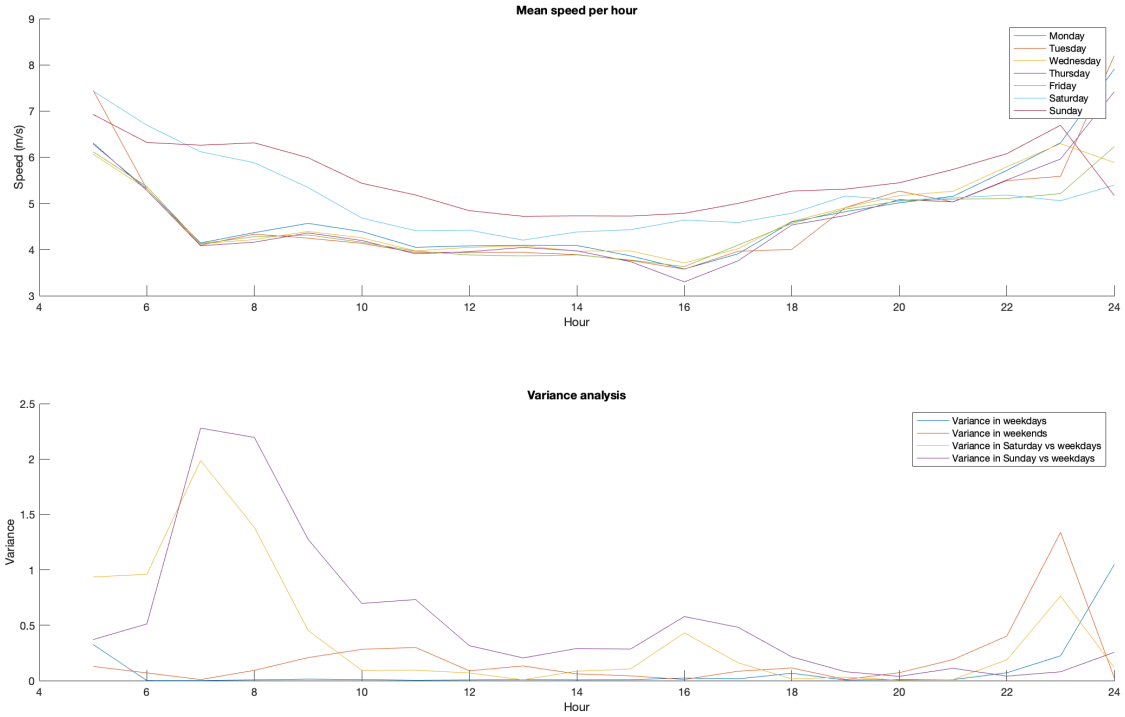
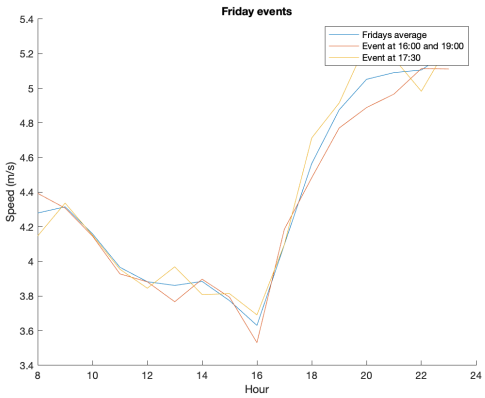


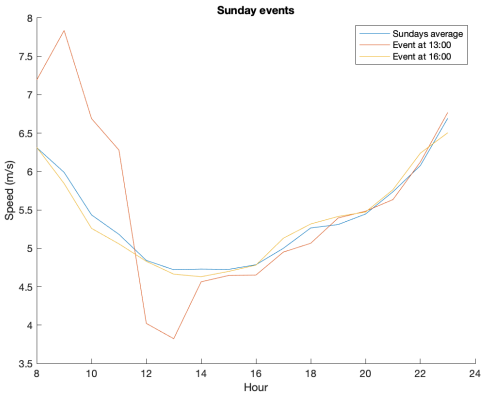
Figure 4.3: Daily speed variation

Due to the chosen scenario area, a list of programmed events at SAAB Arena and Linköping Konsert & Kongress has been collected. However, since the schedule had still to be made at the time, both entities combined could not provide dataset large enough to incorporate these features in the neural network for a more thorough learning process.

In 3.1 the results of the effect on traffic caused by this events are detailed. These results presented are those where the impact of the events was visible, since many didn't reflect any variations respect the usual speed values. In both 4.4a and 4.4b the typical speed of that particular day is displayed in blue and the other lines show the speed of that same weekday when different events were scheduled.



(a) Mean speed during event at 16:00h, 17:30h and 19:00h



(b) Mean speed during event at 13:00h and 16:00h

Figure 4.4

4.3 Implementation and evaluation

4.3.1 Machine learning solution

After the analysis of the different forecasting architectures that might have been able to fit our problem, table 4.1 below summarises the results obtained. The Feed Forward Network is not the best decision when trying to find relations through time since its information flow does can't retain temporal relations. Although Convolutional Neural Networks can have a temporal sense, the architecture performs a better fitting when provided with images as the dataset. Since GTFS feeds do not provide real-time pictures of the road state that option was also discarded. Succeeding this comparison, the decision of using Long short-term memory networks has been made. The framework used has been Keras, an open-source library of artificial neural networks in Python language.

	Non-linear associations	Multivariate data	Parameters auto-update	Time sensitive	Enough data
FFN	✓	✓	✓	✗	✓
CNN	✓	✓	✓	✓	✗
LSTM	✓	✓	✓	✓	✓

Table 4.1: Analysis of four different forecasting models

After deciding the model to be used, two different approaches have been implemented. The first one is a LSTM network which its architecture consists in a layer of four LSTM neurons followed by a dense layer of a single neuron that acts as the output. A dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer,

thus densely connected. The loss function is computed using the mean square error function and uses the Adam optimiser (Kingma and Ba [2014] [18]). The network trains for 1500 epochs, which means that the training data computes this number of one forward pass and one backward passes for each epoch.

The time series data for this case consists in a matrix formed by two columns where the second one acts as the label by containing the speed of the hour that follows the value of the first column. In this case, it is clear that the lag value is set to one since we are forecasting the immediately following value. For this neural network, the dataset has been divided to use 60% as training data and the remaining 40% as test data.

After computing the predictions for the test data, the root mean squared error resulted with the confidence interval of (0.480, 0.539) with a confidence level set to 0.95. When it comes to the mean absolute error the results showed an interval of (0.315, 0.374), with the same confidence. In figure 4.5 the predicted speeds are shown in orange while the ground truths are displayed in blue.

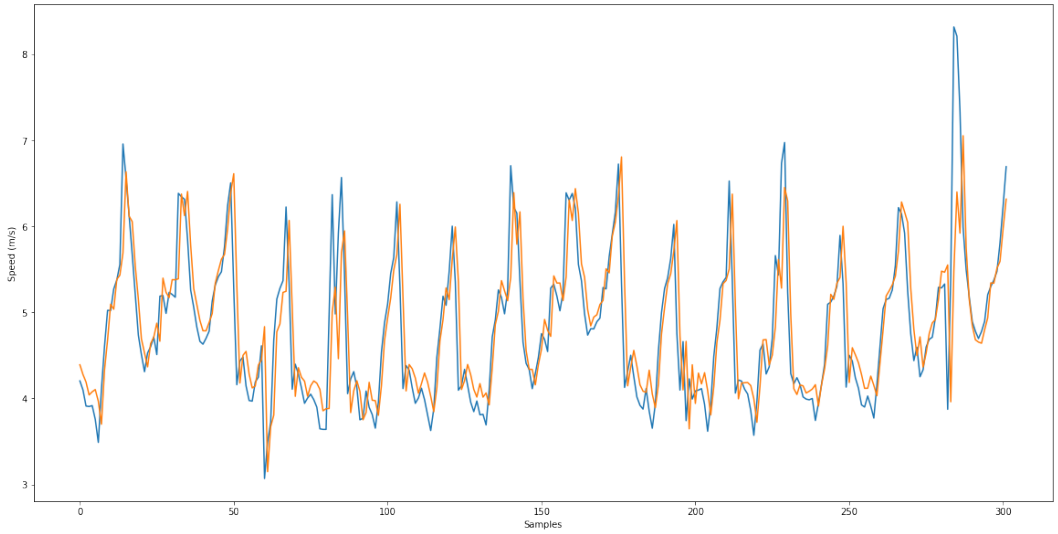


Figure 4.5: One-hour forecast

The second approach consists in predicting several the speed hours forward. In this case, forecasting the following nine hours by looking at the previous 18, which correspond to an entire day. For this network, the input data requires a matrix shaped with 27 columns. Since we look at the values chronologically without missing any, the lag is also set to one. For this case, due to the smaller size dataset after the data processing, initial NaN values are set to zero for the network to learn as starting values. This way, the final dataset ends being a matrix shaped (756, 27).

This network consists in a six neuron LSTM layer followed also by a nine neuron dense layer that acts as output, one for each prediction. The main difference of the LSTM layer from the first approach consist in the input shape and the forecasting values, since the loss function and the optimiser remain the same. The training of this network has been performed with an epoch value of 1000 with one single batch of training data .

Table 4.2 shows the RMSE confidence intervals for each future hour forecast, while table 4.3 shows the same intervals for MAE.

t+1	(0.491, 0.543)	t+4	(0.786, 0.865)	t+7	(1.000, 1.094)
t+2	(0.660, 0.738)	t+5	(0.858, 0.948)	t+8	(1.109, 1.216)
t+3	(0.718, 0.793)	t+6	(0.905, 0.996)	t+9	(1.108, 1.226)

Table 4.2: RMSE confidence intervals for each future prediction

t+1	(0.304, 0.363)	t+4	(0.504, 0.594)	t+7	(0.700, 0.798)
t+2	(0.436, 0.503)	t+5	(0.622, 0.721)	t+8	(0.794, 0.901)
t+3	(0.487, 0.558)	t+6	(0.626, 0.725)	t+9	(0.851, 0.969)

Table 4.3: MAE confidence intervals for each future prediction

In figure 4.6 some samples of the predictions performed by the network are shown in red. Only some of them are displayed for clarity since showing all of them would make it impossible to see the labelled data. The red line consists in the nine different forecasts made after the network receives an input of 18 samples. As it can be seen, the first three hours match with little difference to the ground truth whereas the further in time the prediction are made the error is increases significantly.

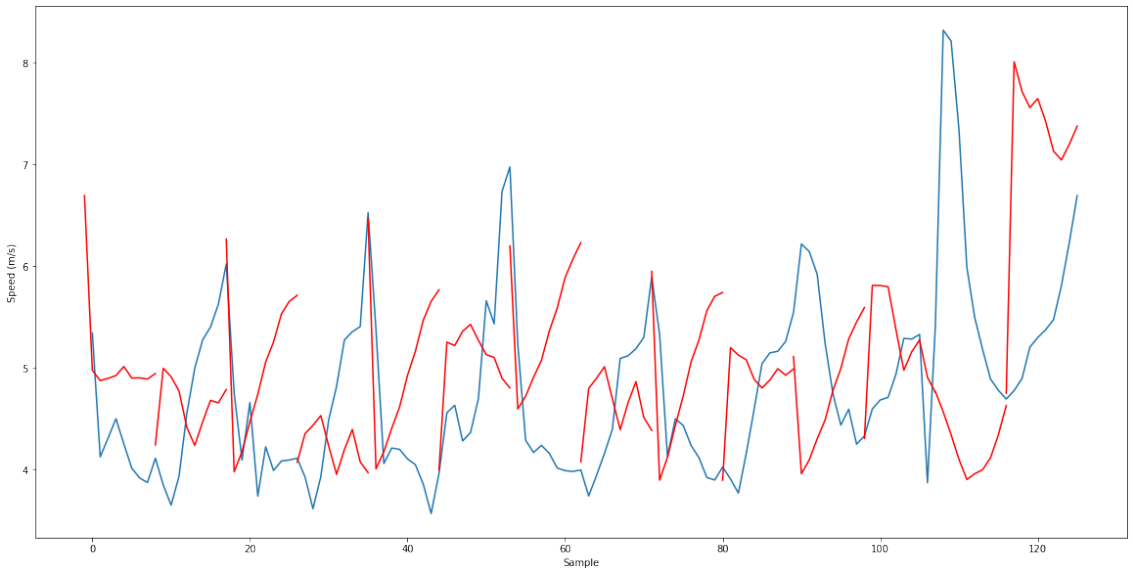


Figure 4.6: Nine-hour forecast

4.3.2 Baseline comparison

After developing both, the machine learning algorithm and the simple statistical model that served as the baseline, the results were plotted and properly compared using the RSME and absolute error errors. In picture 4.7 below we can see the average performed per day that formed the baseline (in orange) used for evaluating the model developed. In blue, we can see the ground truth.

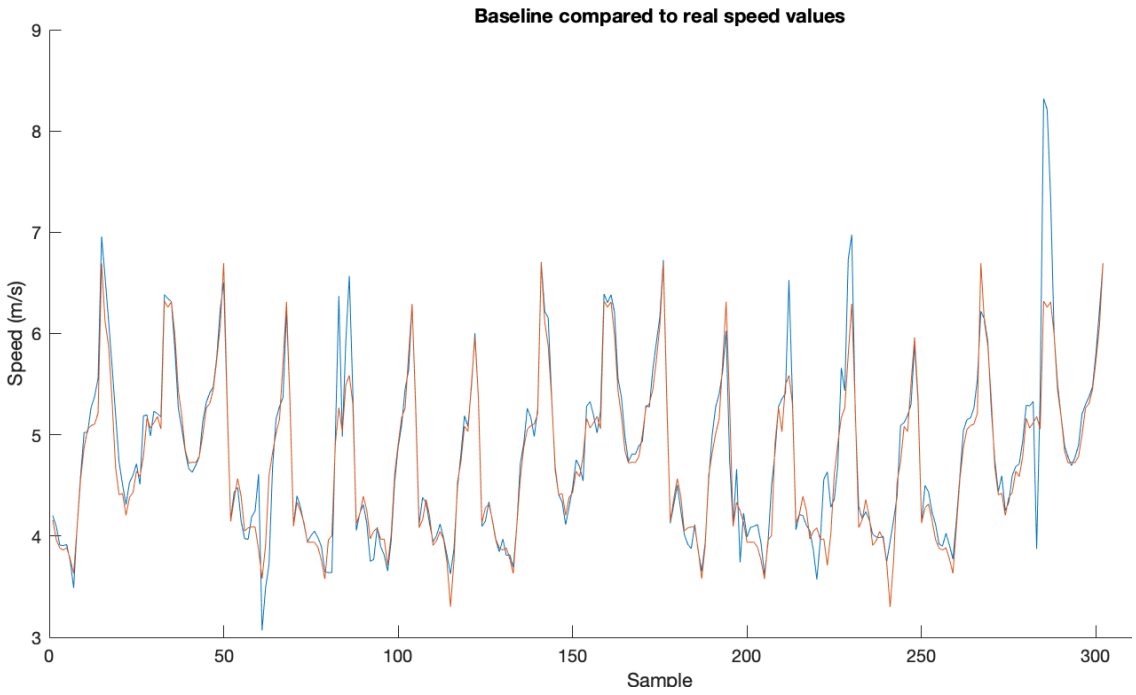


Figure 4.7: Baseline compared to the real speed data collected

When comparing the baseline to the results obtained by the Long Short-Term Memory neural network implementation used in the first approach (where the predictions were made only one hour in the future) the figure below (4.8) was obtained. In blue is displayed the absolute error of the machine learning algorithm while in orange the baseline error. The error is displayed for each of the test data sample. Given the results obtained with the nine hour in the future forecast, this comparison is useful for the first hour prediction. The absolute error comparison between the baseline and the other hours are not shown but it can be easily seen that the forecast perform much worse than the baseline.

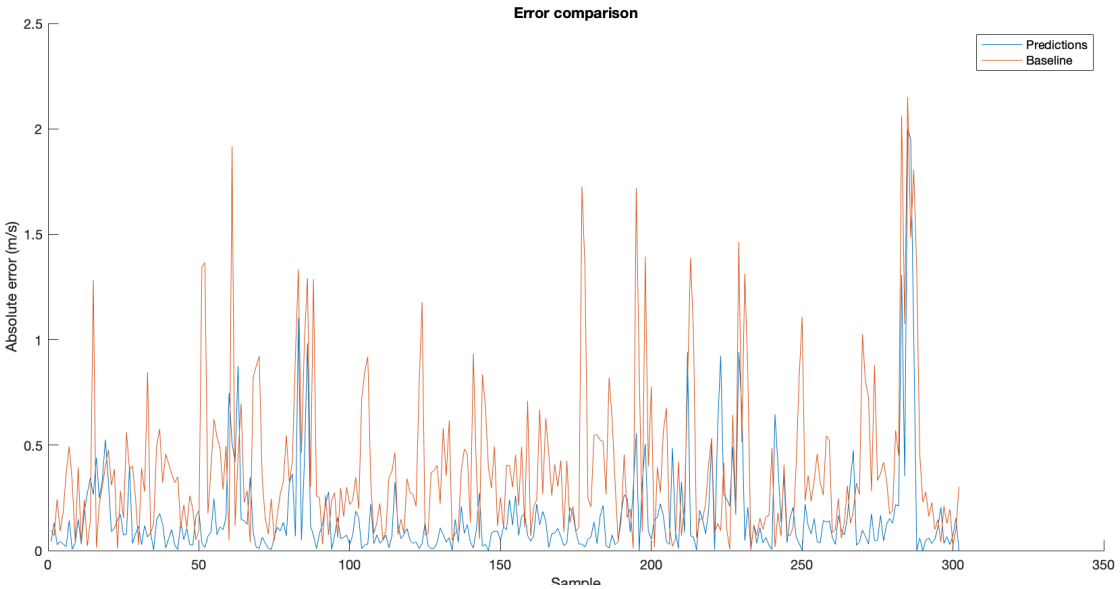


Figure 4.8: Computed baseline and predictions absolute error

5 Discussion

In this chapter the methodology used during the whole project and the results obtained are to be commented. The first part being criticised is the scenario limitations set for the project and the impact that had on the results. Secondly to be discussed are the data used for building the time series and the results given by the output of the LSTM network implementation. Also, the choice of this type of solution and its limitations will be taken into account. Finally, ethical and societal aspects related to the work are going to be studied.

5.1 Results

5.1.1 Scenario

Studying a subarea of Linköping municipality was essential to put some limits to the project. The choice of this scenario 3.1 allowed filtering data that could have compromised the final dataset for being too on the outskirts of the city and always measuring higher values. However, as it can be seen on the heatmap 4.1 it could have also been smaller and more focused on Sankt Larsgatan, the main street when considering the public transport lines. This way, the load of data to process would be smaller and while still containing the low speed values necessary for the congestion labels. It also can be seen that traffic jams do not occur frequently in Linköping, probably because it is a small city with lots of facilities when it comes to other means of transport more environmentally friendly such as cycling, taking the train or the bus. For this reason, conducting this same study in another location with higher density of private vehicles would be interesting to see if there is any variation of the final results.

5.1.2 Data

When it comes to the data used, the main factor to be studied is that it comes from the real-time feeds sent by the local Östgötatrafiken buses. One of the main aspects was the validation that these feeds were representative enough of the traffic state, since speed zero values were reported when the bus approached its stop. It can be seen in the superior plot of 4.3 the 'W' shape of the expected normal speed behaviour with the morning peak relating to the movement of getting to workplaces

and the evening peak of going back home. The values in-between these peaks are slightly low than what was expected and one cause may be the bus stops mentioned before. The values obtained for the daily mean speed of the weekends is higher than weekdays, which is also coherent. When studying the variation between the samples, shown in the inferior plot in 4.3, the only time frame that stands out of the ordinary is the interval from 10:00 PM to 00:00 AM since the only relation with low variance values is when comparing Sundays with weekdays. These differences are neither good nor bad in terms of reflecting the traffic state. However, having these unrelated changes in speed values may lead to more difficulties when the network tries to establish a temporal relation.

In order to make more accurate predictions the attempt of adding other datasets rather than the speed data provided by Östgötatrafiken was made. Even though the data collected was not enough, the events scheduled for the data collection period were only seven, an analysis was made to see if these had a real impact on the city traffic state and if it could be useful to gather a normal size dataset to better the performance of the machine learning algorithm. This impact is displayed in figure 4.4. In 4.4a, the effect is minimum, since there were three events scheduled for that day in either Konsert & Kongress or SAAB Arena and only the ones at 16:00h and 19:00h show a small decrease in speed. However, in 4.4b we can see that there was a programmed event that had a huge impact on the traffic state, since the speed values started decreasing more than an hour before the event at 13:00h and recovered normal values after its start. These results show that the events organised at these venues do not always have an important effect of the traffic flow but they sometimes do, meaning that an updated machine learning solution might be able to learn these relations and extract some conclusions to make more accurate predictions.

5.1.3 Implementation and evaluation

The choice of Long Short-Term Memory networks for developing a solution for this project was sensible after the analysis of the other types of neural networks. In more detail, Feed Forward networks are not able to retain the time related dependencies of the time series fed to the network, an aspect crucial for this specific problem since time is a key factor in traffic congestions. The other algorithm studied, Convolutional Neural Networks, were also discarded since they were mainly developed for image processing and require a hugh amount of data, a bigger one that all the data that was collected during a six week period.

If we analyse the results obtained with the first machine learning approach we can see in figure 4.5 that the prediction matches almost perfectly with the labels of the test data. It is important to notice that the forecast almost fits the real value except in unexpected changes like this last peak. This small underfit can be caused by two main things. The first one is that the network might be unable to learn unprecedented changes and the second one that in this specific time frame there weren't enough data samples to compute an average representative enough of the state. This translates that the network may not be always able to predict sudden high or low changes (real or not) in speed but it is a good predictor for the usual state. If we look at the RMSE or MAE we can see that the values are not really high. In this case, it is more understandable to look at the mean absolute error since it directly translates to error in speed units (m/s), so the in average the error of the algorithm is around 0.345 meters per second, which equals to 1.242 kilometers per hour which is not a significance value when determining if there is a dense traffic state or not.

When comparing this results to the baseline wee can see that the LSTM model is capable of

detecting changes that a normal mean could. Some variations such a as a day or week with a higher traffic influence would not be detected with the simple statistical model since it never varies over weeks. Due to this better adjustments to the real reported values, in general the error of the machine learning algorithm is lower than the baseline as it is displayed in figure 4.8. However, the time limitation did not make it possible but it would be really interesting comparing the solution implemented in this project with a more advanced statistical method to see if it still performs worse since the difference between these errors is not much.

When analysing the second approach, the nine-hour prediction based on the previous 18 hours, the results are coherent: the error increases the further in time the prediction is trying to be made. Although at first sight the results don't seem to be promising, it is important to notice that the error of the first hour prediction remains approximately the same as the one obtained with the first LSTM approach. This means that the network does not need a whole day of samples to predict a single hour.

5.2 The work in a wider context

Using new technologies such as machine learning does not only provide economical benefits but can also contribute to other equally important aspects. Some of the main examples that this project has an impact on are the reduction of local pollution levels where the dense traffic spots are located or the improvement of the efficiency of the local public transport system. By reducing the local levels of pollution, we can help improve the quality of life of the people in the surroundings of the most congested areas. By improving the efficiency of the local public transport system, we might help in the user shifting from private means of transportation to other more sustainable options. When climate change is one of the current threads, small improvements like these can make a difference. Another important aspect is that having a more connected, precise and efficient transport system is key when moving forward to the smart city concept.

Another aspect to consider is the acquiescence of the data used. The feeds provided by Östgötatrafiken come from local buses. This data is not directly linkable with an individual, meaning that anonymity is preserved. Although having more data would make a grate improvement to the results, it always needs to be taken into account its precedence. A balance must be maintained between the amount of data used to obtain the best predictions and the privacy of users. For example, collecting sensory data from private vehicles would be a great source of measures, but it would also mean losing anonymity and an invasion of user privacy.

6 | Conclusion

After the completion of this project some main conclusions can be extracted. The first one relates to the first part of the project: the data acquisition and analysis. The main one is related to the first research question posed at the beginning of this thesis: Is public transport real-time speed data a good representative of the traffic state? Does this data behave as a periodic function?. The answer found is clear, the data reported by the local Östgötatrafiken buses is representative of the traffic state but only in places where the service is provided. This data behaves periodically as what is expected for a normal traffic flow and clearly shows the main differences between daytime and nighttime and also between each day of the week, reflecting the working schedules and the inactivity periods.

The second conclusion is linked with the second part of the project and also to the second and last research question: Can a machine learning algorithm predict the future speed based on previous measurements?. The answer is positive, a Long Short-Term Memory Network can predict future traffic speeds better that a normal statistical average would. However, it has also been seen that that the further in time the prediction is made the greater the error committed, and then using an average to predict the future speed provides a better result than the neural network output.

6.1 Future work

One last important conclusion drawn from this project is that there is still a lot of work that can be made to improve the results obtained. The addition of other datasets for more accurate forecasts is one of the clearest examples. From my experience, trying to collect a near scheduled events dataset as I tried or adding others such as weather conditions or scheduled road works would definitely improve the results. Another option for obtaining more precise forecasts is the enlargement of the speed dataset to enable the auto-update of the network parameters, meaning that the network would be able to learn at the same time that the test data is provided.

Other more general future work aspects to consider would be using this results or even an improved version to implement real city changes such as dynamic public transport lines that avoid areas where traffic jams have been forcasted or with dynamic fares such as discounted prices when the traffic load is high to promote a wider use of public transport as a greener and sustainable way. These data can also be used in other aspects such as city planning for avoiding congestion spots or better understanding of the traffic patterns.

Bibliography

[1] Rajendra K. Pachauri. Climate change, synthesis report. Technical report, Intergovernmental Panel on Climate Change, 2014.

[2] Inc. Google. *General Transit Feed Specification Reference*.

[3] T. M. Mitchell. *Machine Learning*, page 2. McGraw-Hill Science/Engineering/Math, 1997.

[4] Mohssen Mohammed, Muhammad Badruddin Khan, and Eihab Bashier. *Machine Learning: Algorithms and Applications*, pages 7–11. July 2016.

[5] Patrice Simard Yoshua Bengio and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. 1994.

[6] Jürgen Schmidhuber Sepp Hochreiter. Long short-term memory. 1997.

[7] Kenneth C. Lichtendahl Jr. Galit Shmueli. *Practical Time Series Forecasting with R: A Hands-On Guide*, pages 18–19. Axelrod Schnall Publishers, July 2015.

[8] Paul Werbos. Beyond regression: new tools for prediction and analysis in the behavioural sciences. 1974.

[9] Gábor Petneházi. Recurrent neural networks for time series forecasting. 2019.

[10] Nicol N Schraudolph Felix A Gers and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, August 2002.

[11] Gautam Shroff Pankaj Malhotra, Lovesh Vig and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. Presses universitaires de Louvain, 2015.

[12] Paratapa Goswami Éric Gaussier Ali Ait-Bachir Yagmur Gizem Cinar, Hamid Mirisaee and France Vadim Strijov. Time series forecasting using rnns: an extended attention mechanism to model periods and handle missing values. 2017.

[13] Dihua Sun Min Zhao, Xi Chen and Ton Zhou. An algorithm for freeway traffic stare detection considering speed difference characteristic. June 2016.

[14] Junjie Wu Guannan Liu Zhan Xiong Jinguyan Wang, Qian Gu. Traffic speed prediction and congestion source exploration. December 2016.

[15] Gtfs regional (beta). `https://www.trafiklab.se/api/gtfs-regional-beta`.

[16] Pandas data analysis library. https://pandas.pydata.org.

[17] Panayotis Christidis and Juan Nicolás Ibáñez Rivas. Measuring road congestion. 2012.

[18] Jimmy Ba Diederik P. Kingma. Adam: A method for stochastic optimization. 2016.