DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2018

An evaluation of deep neural network approaches for traffic speed prediction

COSAR GHANDEHARIOON

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

An evaluation of deep neural network approaches for traffic speed prediction

COSAR GHANDEHARIOON

Master's program in Software Engineering of Distributed Systems
Date: January 16, 2019
Supervisors: Ian Marsh and Ahmad Al-Shishtawy
Examiner: Sarunas Girdzijauskas
Principal: RISE SICS AB
School of Electrical Engineering and Computer Science

Abstract

The transportation industry has a significant effect on the sustainability and development of a society. Learning traffic patterns, and predicting the traffic parameters such as flow or speed for a specific spatiotemporal point is beneficial for transportation systems. For instance, intelligent transportation systems (ITS) can use forecasted results to improve services such as driver assistance systems. Furthermore, the prediction can facilitate urban planning by making management decisions data driven.

There are several prediction models for time series regression on traffic data to predict the average speed for different forecasting horizons. In this thesis work, we evaluated Long Short-Term Memory (LSTM), one of the recurrent neural network models and Neural decomposition (ND), a neural network that performs Fourier-like decomposition. The results were compared with the ARIMA model. The persistent model was chosen as a baseline for the evaluation task. We proposed two new criteria in addition to RMSE and r^2, to evaluate models for forecasting highly variable velocity changes. The dataset was gathered from highway traffic sensors around the E4 in Stockholm, taken from the "Motorway Control System" (MCS) operated by Trafikverket.

Our experiments show that none of the models could predict the highly variable velocity changes at the exact times they happen. The reason was that the adjacent local area had no indications of sudden changes in the average speed of vehicles passing the selected sensor. We also conclude that traditional ML metrics of RMSE and r^2 could be augmented with domain specific measures.

Keywords: Deep Learning, Regression, Time Series, LSTM, Neural decomposition.

Sammanfattning

Transportbranschen har en betydande inverkan på samhällets hållbarhet och utveckling. Att lära sig trafikmönster och förutsäga trafikparametrar som flöde eller hastighet för en specifik spatio-temporal punkt är fördelaktigt för transportsystem. Intelligenta transportsystem (ITS) kan till exempel använda prognostiserade resultat för att förbättra tjänster som förarassistanssystem. Vidare kan förutsägelsen underlätta stadsplanering genom att göra ledningsbeslut datadrivna.

Det finns flera förutsägelsemodeller för tidsserieregression på trafikdata för att förutsäga medelhastigheten för olika prognoshorisonter. I det här avhandlingsarbetet utvärderade vi Långtidsminne (LSTM), en av de återkommande neurala nätverksmodellerna och Neural dekomposition (ND), ett neuralt nätverk som utför Fourierliknande sönderdelning. Resultaten jämfördes med ARIMA-modellen. Den ihållande modellen valdes som utgångspunkt för utvärderingsuppgiften. Vi föreslog två nya kriterier utöver RMSE och r2, för att utvärdera modeller för prognoser av högt variabla hastighetsändringar. Datasetet insamlades från trafiksensor på motorvägar runt E4 i Stockholm, för det så kallade motorvägskontrollsystemet (MCS).

Våra experiment visar att ingen av modellerna kan förutsäga de höga variabla hastighetsförändringarna vid exakta tider som de händer. Anledningen var att det intilliggande lokala området inte hade några indikationer på plötsliga förändringar i medelhastigheten hos fordon som passerade den valda sensorn. Vi drar också slutsatsen att traditionella ML-metrics av RMSE och R2 kan kompletteras med domänspecifika åtgärder.

Nyckelord: Djupinlärning, Regression, Tidsserier, LSTM, Neural dekomposition.

Acknowledgement

This project is conducted as my master thesis at KTH and is a part of the BADA project at RISE SICS AB. I would like to thank my examiner Sarunas Girdzijauskas and my supervisors at KTH, Vladimir Vlassov, Ahmad Al-Shishtawy and Zainab Abbas. Their comments and feedbacks enriched this work and helped me to learn a lot during the project. Special thanks to Ian Marsh my supervisor at RISE SICS, without his guidance, invaluable insights and kind support it was not possible for me to finish the project successfully. My family is always my source of energy to go forward, I want to thank them all for being there for me.

Contents

List of figures

List of tables

1. Introduction

Traffic congestion causes several problems such as extra travel time for the commuters, increase fuel consumption and transportation cost. From an environmental perspective, air pollution is increased from car emissions in traffic jams. Congestion can be sometimes reduced by adding more capacity to the traffic network. However, building infrastructures are difficult and expensive and raise environmental issues. Another solution to the congestion problem is to decrease the demand for using the congested area during rush hours.

Predicting average speed can help us to take proactive action and reduce congestion by providing information for drivers and driving assistance systems to have smarter travel planning [23]. Providing more information can also improve public transport services. The arrival time of buses can be estimated by using predicted velocity in real-time. Realistic schedules encourage citizens to use public transport and decrease using private cars, thus reduce congestion.

Many deterministic or stochastic models have been investigated to predict the average speed of vehicles. In most cases, a centralised system carries out the prediction using the global knowledge of a traffic network [9-11]. The centralised system gathers information from all the sensors in a traffic network and predict congestion. However, they are usually too time-consuming to be used for real-time usage. If a prediction model, can forecast the velocity from a subset of local sensors, the model can be used for real-time traffic flow. The information can be sent to the drivers that are planning to pass the congested area and enables them to select better routes along a less congested path.

There are several models such as ARIMA, the autoregressive integrated moving average model that have been used to predict traffic parameters over many years [4]. Neural networks can predict the behaviour of a system by learning the traffic flow and speed from pre-observed historical data. Moreover, there are new techniques that are now being used in classification and regression problems. One of the prominent techniques is deep learning. Deep learning is a class of machine learning technique applied to different areas such as computer vision,

bioinformatics, finance amongst other areas [1]. Frameworks like Scikit Learn, TensorFlow and Keras have been developed to design, implement and evaluate prediction models.

Given previous prerequisites, the question posed in this thesis is:

"Is it possible to predict the future average speed of vehicles at a specific spatiotemporal point by using local neighbourhood knowledge, furthermore with acceptable accuracy and performance?"

1.1. Problem description

We need to find a prediction model for a traffic point that can forecast the average speed of vehicles, based on local neighbourhood knowledge at short, medium and long term forecasting horizons. Given the overall goal above, we can break that overall goal to several sub-goals:

Q1. Is it possible to predict the average speed at a geographical point independently, accurately and efficiently? Do added upstream and downstream points improve the prediction?

Q2. Does the model perform sufficiently in predicting highly variable velocity changes prediction?

Q3. Is the model able to outperform traditional methods for different forecasting horizons? Are there any improvements based on the selected criteria, namely RMSE, r^2, transition ratio and accuracy?

Q4. How does the time aggregation of the input data affect the prediction results? Does it have a positive or negative effect? By time aggregation we mean taking averages from the original 1 minute samples (which are already time-averaged) to 5 minutes.

1.2. Thesis objective

The goal of this project was to find a model that can predict the average vehicle speed based on historical data of neighbouring nodes. The model should have acceptable accuracy and be able to perform well in learning highly variable changes.

1.3. Ethics and Sustainability

Ethics is a crucial aspect of any scientific research. The dataset used for this work was gathered by the Swedish transport administration agency (Trafikverket) and has been used, with permission, by RISE SICS for research purposes in the BADA (Big Data Analytics for Automation)[1] project.

This work is a part of the project and RISE has legal access to the dataset. Besides, the usage of tools and APIs complies with their terms of service. Furthermore, this thesis supports the software engineering code of ethics and professional practice performed by the ACM / IEEE-CS joint task force, that states *"software engineers shall ensure that their products and related modifications meet the highest professional standards possible"*.

From a sustainability perspective, transportation system plays a vital role in sustainable improvement. This research aims to find an accurate and efficient prediction model for traffic parameters. The result of this research can facilitate travel planning for commuters. Additionally, it can improve driving assistance systems services to find a less congested path to the desired destination. Besides, it can help authorities to make data-driven decisions for urban planning which is beneficial for all the citizens.

1.4. Research methodology

In this work, we used an applied research method. The applied research method applies existing research materials to real-world data to solve problems [15]. Our

[1] http://bada.sics.se/

problem in this thesis was to evaluate models that predict the average speed at a specific spatiotemporal location for short, medium and long time horizons. We used the Autoregressive Integrated Moving Average (ARIMA), Long Short-Term Memory (LSTM) and Neural decomposition prediction models. The real-world dataset was the traffic data recorded by the Motorway Control System (MCS) in the Swedish transport administration.

Besides, We used quantitative research for evaluating prediction models. We measured the performance of the selected models based on various numerical criteria namely RMSE, r^2, transition ratio and accuracy.

1.5. Thesis scope

This thesis evaluated ARIMA, LSTM and Neural decomposition models for short (time), medium (time) and long forecasting horizons.

The structure of the models is explained in the background section. A limited input and output structures were used to train the models. A time-aggregated dataset was created from the input data to observe the effect of time aggregation of the inputs. Essentially is averaging data more likely to produce better ML results? The dataset was a real traffic data from physical sensors located on the highways around Stockholm, namely the E4.

The input data was the average speed of vehicles from traffic points every minute. The metrics used for evaluation were the root mean square error, the coefficient of determination, transition ratio and accuracy. The performance evaluation of each forecasting model was based on the time elapsed for training and prediction. Some topics excluded from this work were the partitioning the transportation system into smaller subsections, that was studied in Reginbald [14], community detection for partitioning the road graph investigated by Thorri [13]. Other machine learning methods namely, decision trees, random forests and logistic regression have been covered by Consuegra in [12] for anomaly classification.

1.6. Structure of the thesis

The background work is presented in chapter 2 and contains the details of the forecasting models. Chapter 3 contains the methodology of the thesis work. The steps of pre-processing, models' implementation and selected evaluation measures are also described in this section. The results of the experiments are presented in chapter 4. In chapter 5, the interpretation of the results is discussed. The limitations are stated in chapter 6. In chapter 7, recommendations for Trafikverket is stated. Chapter 8 presents the conclusion of the experiments. In chapter 9 the lessons learned during the project are presented. Future works are suggested in chapter 10 as a guideline for further studies.

2. Background and related work

The relevant theory is provided in this chapter for understanding the rest of this work. The background section contains key terms and concepts including a detailed description of the selected forecasting models. The previous works are reviewed including previous studies and experiments for traffic prediction.

2.1. Background

This section presents the basic concepts and specifications of the time series. Moreover, the structure of the prediction models are explained, and the necessary mathematical equations are described. Stated prediction models are the statistical ARIMA model, Neural networks, Deep neural networks, Recurrent neural networks, Long short-term memory and Neural decomposition.

2.1.1. Time Series

Time series as defined in [4, chapter 2] is a set of values in time order. If there is only one value at each time step, the time series is known as a univariate one. Values of the time series can be discrete or continuous [4]. If the time series contains multiple values at each step, it is known as a multivariate time series. Therefore, preserving the order is important when it comes to *learning* their past behaviour. There are several examples of time series, like weather temperatures, sales data, flight passengers throughout the year and so on.

Time series in general consist of four components [3, chapter 6]. A trend component, a cyclic component, a seasonal and residual components. The trend is the general direction of a time series in long-term. It can be growing, decreasing or staying steady over a long period. The cyclic variation shows the repetitive medium-term changes that do not have a fixed length, especially in business and finance. Seasonal components are fluctuations in a yearly period within a season. Remaining components are the random variation, the residual after removing trend, cyclic and seasonality. They could happen irregularly and therefore can be hard to predict.

Many approaches have been taken to model time series and predict values based on history. As described in [4, chapter 3], stochastic models like ARIMA, Artificial neural networks (ANN), Support vector machine (SVM) are commonly used methods for classification and regression of time series. A predictive regression model should observe the historical data and approximate a function to map input data to a continuous output quantity.

2.1.2. Autoregressive integrated moving average model (ARIMA)

Autoregressive integrated moving average models (ARIMA), are one of the most known statistical prediction models commonly used for forecasting time series. It is a combination of Autoregression (AR), Integration (I) and Moving average (MA) operations.

Autoregression (AR) as defined in [22] is "a model that uses the dependent relationship between observation and some number of lagged observations" . In an autoregression model, the estimator predicts the value using a linear combination of previous values of a variable. AR(p), an autoregressive model to compute the variable y at time t with p lags, is as follows:

$$y_t = \mu + \sum_{i=1}^{p} \gamma_i \, y_{t-i} + \varepsilon_t \tag{1}$$

In equation (1), μ is a constant and γ_i is a coefficient for y_{t-i} the lag variable at time t-i and ε_t is the error term at time t.

The Moving Average (MA) forecast model is "a model that uses the dependency between an observation and a residual error from a past few forecast values" [22]. MA(q), a moving average model to compute variable y at time t with q lags, is as follows:

$$y_t = \mu + \sum_{i=1}^{q} \theta_i \, \varepsilon_{t-i} + \varepsilon_t \tag{2}$$

In equation (2), θ_i is the coefficient for the lagged error term at time t-i.

The integrated (I) model, "uses differencing of the raw observations to make the time series stationary" [22]. For instance, subtracting an observation from observation at the previous time step is I(1). The equation of I(d) for the variable y_t integrated of order d, is as follows:

$$\Delta y_t = y_t - y_{t-d} \tag{3}$$

As stated in [22], if we combine differencing, autoregressive and moving average models, an ARIMA model is the result. ARIMA(p,d,q) is a model with p autoregressive lags, q moving average lags, and a difference of order d.

2.1.3. Neural networks

A neural network is a simplified model of the biological structure in the human brain. It consists of several neurons arranged in layers which are connected in chains. Information flows into a neuron and a result flows out from it. As shown in figure 1, there is always one input layer, one output layer, and one or more middle layers which are also called hidden layers. The output of each layer is the input of the next layer, therefore this configuration is called a feedforward network.

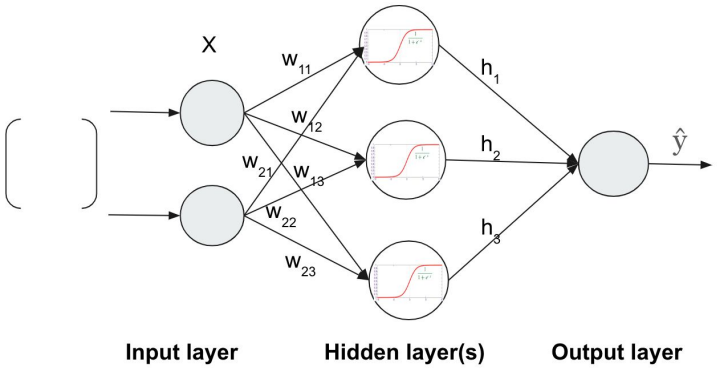


Figure 1. A schematic representation of a neural network, information flows into a neuron and a result flows out.

The input to each neuron is a combination of the weighted values of the previous layer outputs, i.e. a linear summation. Each neuron has an activation function that is a nonlinear function that maps the value of the sum to [0,1]. The output of the activation function results in activating or not the neuron. Examples of activation functions are sigmoid, relu, tanh. By executing the nonlinear function to the combined input value, an output value of the neuron is produced. If it is zero, then the neuron is not activated otherwise, it is and it passes the computed value to the next layer.

Another term used in neural networks is bias, that is a scalar added to the sum before using the activation function. The neurons are activated when the weighted sum is higher than bias value. The output of a hidden layer is a sum of weighted inputs plus the bias.

A matrix representation of the input, weights and biases of the network in figure 1 is as follows:

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad w = \begin{pmatrix} w_{11}\ w_{12} \cdots w_{1n} \\ w_{21}\ w_{22} \cdots w_{2n} \\ \vdots \\ w_{m1}\ w_{m2} \cdots w_{mn} \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \tag{4}$$

The output of the hidden layer h is computed as stated in [1, chapter 6] with the following equation, g is the activation function as mentioned above.

$$h = g\,(w^T x + b\,) \tag{5}$$

The output of each layer is computed based on the weights and biases; these values are initiated with random values. For a learning task, they should be updated based on the results of computations. Therefore a cost function or error function is defined to compute the *distance* between the result of manipulation and the actual value it should produce. As stated in [1, chapter 4] For proper learning this cost function should be minimised that is called *optimisation*. For

minimising the cost function, the derivative of the cost function should be minimised. Since all the weights and biases are represented in matrix form, the derivative of the cost with respect to the weights and biases is the *gradient* matrix, and the optimisation method is called *gradient descent optimisation*.

Ian Goodfellow et al. in [1, chapter 6] described that the *forward propagation* is the regular computation flow from x as input to $\hat{y}$ as output, shown in figure 1. The forward propagation can continue to compute the cost value. Then the information from the cost value can flow backwards through the network to compute the gradient. This is called *backpropagation*. The backpropagation increases the accuracy of the model and enables it to learn from previous error values. Back propagation is the actual learning process of the model.

Simply the goal is to find biases and weights that *minimizes* the error function.

2.1.4. Deep neural networks (DNN)

Deep neural networks is a neural network with many hidden layers. Deeper layers allow the model to learn different features at each layer. One of the advantages of deep neural networks is that they eliminate feature engineering of classical neural networks [1]. The advancement of technology in distributed computing and powerful hardware units specifically GPUs has enabled and advanced DNNs. They perform many matrix operations that was not reasonably possible before.

There are some specialised models in deep neural networks, one of them is convolutional neural networks (Conv. nets) that are used practically in the commercial applications of the neural networks [1, chapter 9]. Conv nets are fitted for matrix structured datasets and have been successful in image classification. Another specialised model that is used in one-dimensional sequential data is recurrent neural networks.

2.1.5. Recurrent neural networks (RNN)

Recurrent neural networks are a recursive network that has the advantage of sharing parameters between computations in successive time steps. This property enables them to learn patterns from longer sequences compared to regular neural

networks. As described in [1, chapter 10], The recurrence is a connection from the output to the hidden layer. Essentially, the RNN unit has a *state* which is updated when it generates an output and is used for computing in the next time step. Therefore the output is a combination of input and the state from the previous computation. Figure 4 illustrates the schematic of one RNN unit.

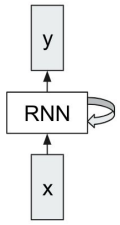


Figure 2: representation of one RNN unit, x is the input vector, and y is the output vector.

The recurrent equation of the output computation of an RNN unit at each time step is as follows, h_t is the new state, g_w is the function with parameters w, h_{t-1} is the old state, and x_t is the input vector at time step t.

$$h_t = g_w(h_{t-1}, x_t) \tag{6}$$

The significant aspect of RNNs is that they provide flexibility regarding input and output structures. They allow vector of sequences as input and vector of sequences as output and even using both for more complex tasks. Figure 3 taken from Andrej Karpathy blog post [6], shows some examples of these models.

The example in [6] for one-to-many transformation is image captioning, a vector of images as input and sequences of words as output. Sentiment analysis is an example of many-to-one transformation is where a sequence of words in a text is the input, and the positive or negative expression is the output. Many-to-many examples are machine translation tasks, where a sequence of words as the input and the translated sentence the output.

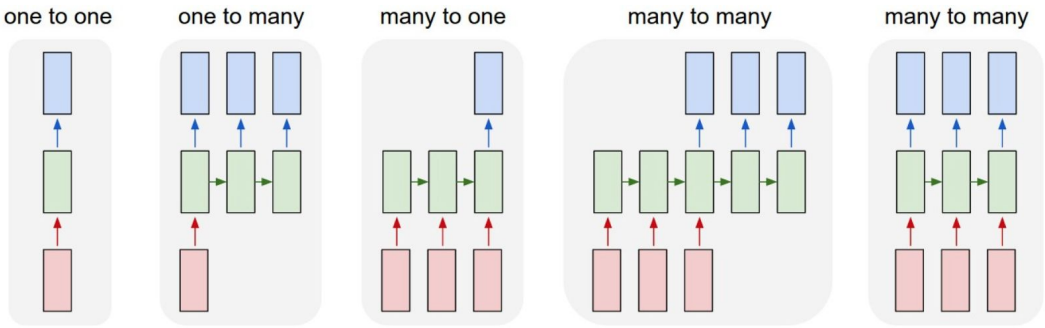


Figure 3: Examples of Recurrent Neural Networks, red boxes are input vectors, green boxes are RNN units holding cell state, and blue boxes are output vectors. The arrows are functions e.g. matrix multiplication, see [6].

For backpropagation, the gradient is computed recursively starting from the final loss. The computational challenge here is that the gradients propagated in many steps will be vanished or exploded. As stated in [1, chapter 10] whenever the model shows the long-term dependency, the gradient is exponentially smaller than the short-term dependencies. The signals from long-term dependencies are hidden by signals coming from short-term dependencies. There were several approaches to handle this problem and the most effective one is called gated RNNs. Basically, the model learns when to forget the old state.

2.1.6. Long short-term memory (LSTM)

As stated in the previous section, recurrent neural networks have the vanishing gradient problem, and gated RNNs have the best solution to address this problem. Long short term memory LSTM, is a gated RNN. The aim is to create paths through time to avoid vanishing or explosion in gradient. The recurrence equation in LSTM is more complicated than a simple RNN unit.

LSTM unit is shown in figure 4 from the Olah C. blog post [5]. The LSTM gated structure enables it to control the mechanism of updating the cell state. Merely, the gates decide when and what information passes through the cell.

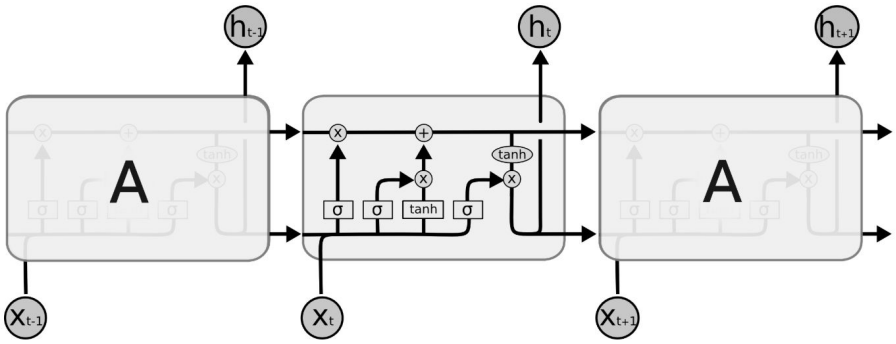
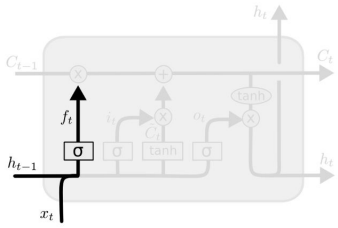


Figure 4: LSTM Unit, the top horizontal line is the cell state, X is input and the combination of sigmoid function (σ) and the pointwise multiplication (⊗) shows the gates [5].

The gates are the combination of sigmoid function (σ) and the pointwise multiplication (⊗) as shown in figure 4. The first one on the left is the forget gate, the second one combined by tanh applied on the input is the input gate and the last one combined by tanh applied on the cell state is the update gate. The reason to use sigmoid function is that the sigmoid generates values between zero and one, if the input is zero, means that nothing should pass through, and if it is one, it means that everything should be passed.

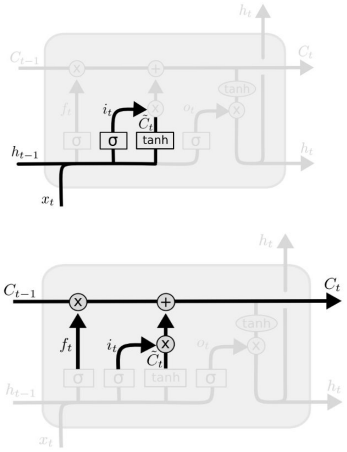
The first decision for the LSTM unit is what information should be thrown away and the *forget gate* does it. The sigmoid says keep this state or forget it. Figure 5 [5], shows the forget gate equation.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right) \tag{7}$$

Figure 5: Forget gate [5].

The next decision is what information should be stored in the cell state. In the *input gate*, the sigmoid decides on what should be stored and the Tanh creates a vector of candidates. By combining these two functions, the new value of the state is decided. Figures 6 [5] shows the input gate equations.



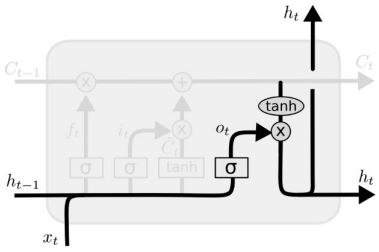
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

(8)

Figure 6: Input gate, C_t is the cell state [5].

The last decision is what part of the cell state should be sent as an output and this is done in *update gate*. The sigmoid decides on the parts that should be filtered and the parts that should be sent to the output. Also, the Tanh pushes the value between 1 and -1. The combination results in the output value. The update gate's equation and the output of the hidden layer equation *h* are shown in figure 7 [5].



$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

(9)

Figure 7: Update gate, h_t is the output of the hidden layer [5].

There are several different variations of LSTM as described by Klaus Greff et al. in 'LSTM, a search space odyssey' [7]. They studied different models based on different combination of gates to see which one had the most effect on the results. No input gate (NIG), no forget gate (NFG), no output gate (NOG), no input activation function (NIAF), no output activation function (NOAF), coupled input and forget gate (CIFG) referred to as gated recurrent units (GRU), no peepholes (NP) and full gate recurrence (FGR) models were implemented. Afterwards, they applied all these models on three datasets of speech, handwriting and the chorales. They argued that the forget gate and the output activation function were the essential components of LSTM and concluded that none of these variants could perform significantly better than the standard LSTM.

2.1.7. Neural decomposition

Luke Godfrey et al. in [8] explained a model of feedforward neural network that fits a curve to the sample data and then used the trained curve to forecast values. They argued that if we assume a time series as a signal, then it is possible to apply Fourier transformation on it and decompose it to the sum of sinusoid components. This approach can help a model to learn the non-linear periodic pattern of historical data and enables us to make a prediction based on the components. The problem of Fourier transformation is that it can only learn the periodic behaviour of time series and the linear behaviour such as trend will be missed. Therefore, in their proposed network model, they added a few linear layers to let the model learn the non-periodic components at the same time.

Neural Decomposition (ND) proposed in [8], is constructed from a layer of neurons using *Sinusoidal activation function* to learn periodic behaviour and a layer of neurons using *augmentation function* to learn the non-periodic behaviour of a time series. The augmentation function is composed of different activation functions, *linear* function to learn the trend, *Sigmoid and Softplus* function to learn non-periodic and nonlinear components. They train the model using *stochastic gradient descent* technique with *backpropagation* for the learning task. Figure 8 driven from [8], shows the schematic representation of Neural decomposition network.

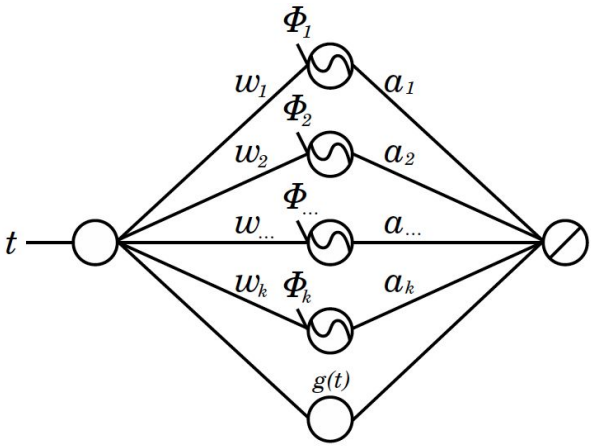


Figure 8: The diagram of the ND neural network, for each sinusoid unit w_i is the frequency, Φ_i is the phase shift and a_i is the amplitude, g(t) is the augmentation function [8].

The model as defined in figure 8, has k neurons with sinusoid activation function. For each neuron, weight w_i is the frequency, bias Φ_i is the phase shift and a_i is the amplitude, g(t) is the augmentation function that learns the linear component. An output layer is a single unit that combines all periodic and non-periodic components of the hidden layer. The equation of the output x(t) in the ND network is as follows:

$$x(t) = \sum_{k=1}^{N} \left(a_k \cdot sin(w_k t + \phi_k)\right) + g(t) \tag{10}$$

2.2. Related work

In this section, previous works for traffic parameters prediction using neural networks are reviewed. Then a discussion is presented to show the comparison of what is investigated previously in the paper and what is done in this thesis work.

2.2.1. Traffic parameter prediction papers

Fouladgar et al. in [9] proposed a decentralised deep learning based method to predict traffic congestion at each traffic point based on local measurement of the neighbouring points. Two deep learning models were implemented to predict the traffic flow of southern California, based on the traffic pattern learned from northern California. They used Convolutional neural network (CNN) and Long short-term memory (LSTM) network for prediction tasks. In CNN, the inputs are traffic condition and incidents information for all geographical points. They focused on junctions (intersections, exits or entrances) as traffic points, and model the roads as a network of connected points. For each network point inflow and outflow sequence based on four adjacent network points (2 previous and two next points) are obtained. The input parameter for each position is the ratio of the average speed of vehicles to the speed limit at each traffic point, that is a value between 0 and 1. Incidents are weather condition, car accident information, road construction, holiday dates and events like matches or concerts. Incident information is added through a fully connected layer. In LSTM, four last time points are selected at 5-minute time intervals. The Euclidean loss function is applied to train the network and is regularised in favour of high congestion samples over low congestion to avoid biased training.

In another work by Thomas Epelbaum et al. in [10], the authors evaluated three neural network models to forecast the speed of vehicles. They applied FNN, CNN, and LSTM on traffic data. The dataset was the speed of every 3 minutes of the external ring road of Paris. They divided raw speed by free-flow speed (65 km/h) to normalize it. Two input structures were used, the original dataset and the reduced dataset. The reduced dataset was the average over five-time intervals, that was 15 minutes. The root mean square error RMSE was used to assess the quality of the model's prediction. Also, they proposed Q2 score to compare RMSE with 'Real-time propagation benchmark (RTPB)'. It was concluded that FNN and RNN outperform CNN for on the dataset.

Yisheng Lv et al. in [17] used a stacked autoencoder (SAE) model to learn the traffic flow features, the training set was used as an input to the first autoencoder layer. Afterwards, the output of each hidden layer was used as the input of the next autoencoder layer. On top of the last layer, a logistic regression layer was added for traffic flow prediction task. Their dataset was the flow data every 30 seconds from more than 15000 detectors across California. They aggregated the data for 5-minute time intervals before training the model. The SAE model was compared with the random walk (RW), back propagation (BP) NN, support vector machine (SVM), and the radial basis function (RBF) NN model. For the evaluation task, the mean absolute error (MAE), the mean relative error (MRE), and the root mean square error (RMSE) indexes were used. Four forecast time horizons were selected, 15-minute, 30-minute, 45-minute, 60-minute. The grid search was used to obtain the best architecture for different prediction tasks. The authors discussed that the SAE model, unlike the other studied methods could discover the nonlinear spatiotemporal correlations in traffic data.

Yanjie Duan et al. extended their previous work [17] in [18] by performing several experiments on their SAE for the effect of evaluation measures during day and night. They concluded that MAE and RMSE during the day are larger than that during the night. While the MRE during the day is smaller than that during the night. They concluded that, for traffic flow prediction, using SAE on a real dataset, a combination of multiple models for different time span should be used.

2.2.2. Relation to this work

What was done by Fouladgar et al. In [9], has some similarities and differences to our work. They focused on traffic data from the sensors of a region and proposed a decentralised approach. We also focused on the information of the neighbourhood and not the whole network to have an independent prediction model. They used Convolutional neural network to compare with the Recurrent neural network for prediction, while we did not consider CNN as they were not well suited for the sequential data as they concluded.

In Epelbaum work [10], FNN, CNN, RNN were used to predict the speed of cars, while we used LSTM that is a promising model for time series. Also, they used a

relative parameter that is a ratio of speed over the free flow speed of the road. We studied the average speed to avoid computed parameter.

In Yisheng's work, the whole network of detectors was considered as the input dataset, whilst we focused on local traffic data. They studied the flow parameter while we focused on the average speed parameter. We both used a 5-minute aggregated dataset. They stacked autoencoders to build a deep neural network, while we used LSTM units for learning nonlinear spatiotemporal correlation. One of the evaluation measures (RMSE) was similar to what we used. The random walk (RW) method, used as the baseline model, was similar to our persistent baseline model.

3. Methodology

As shown in figure 9 different classes of models were chosen for time series regression prediction. In class A, prediction models use a rolling technique to forecast future samples. Few previous lagged value are made available for the model in a time window, then the first time interval is predicted and the time window is shifted forward for the next time step. ARIMA was chosen from class A. Class B are the recurrent models that share parameters between successive time steps. LSTM was selected from class B. Class C are the models that use regression-based extrapolation. They fit a function to a training data, then use the trained function to anticipate the future samples. Neural decomposition was a representative of class C. Persistent model and the previous week observation model were also implemented to have a naïve model to use as a baseline for the evaluation task.

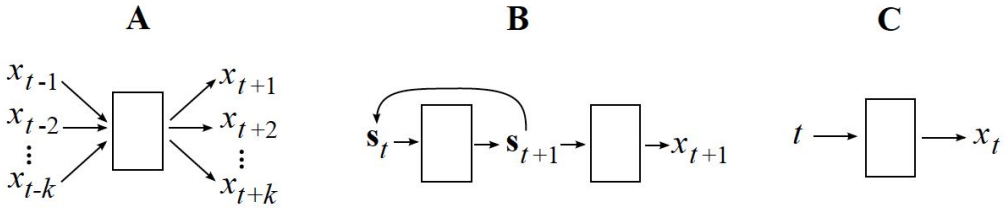


Figure 9: classes of prediction model [8]. ARIMA, LSTM and ND are the representatives of the classes A, B and C respectively.

3.1. Forecast models

There are several predictive models for time series forecast as described by Adhikari et al. in [4]. In this thesis, we selected the following models based on the background studies and related work reviewed in sections 2.1 and 2.2.

3.1.1. Persistent model

For any evaluation task, there should be a naïve model to use as a baseline in the comparisons. The performance of all other models should be above the baseline to be considered as a proper solution for the prediction task. A good baseline model should have three essential properties as explained by J. Brownlee in [16]. First, it

should be straightforward to compute and easy to implement. Second, there should be no intelligence or learning. Third, it should be consistent, that means it should always give the same result for the same input if it is run several times. For this work, a persistent model was implemented, that uses the last observed value as a predicted value for the next step.

3.1.2. The previous week observation model

Another naïve model was the previous week observation, that assumes the previous week observed value for a time step as a prediction for this week. The persistent model was implemented because it is a commonly used baseline for prediction tasks, and the previous week observation was implemented since a weekly pattern was observed on the average speed in the dataset.

3.1.3. ARIMA

ARIMA is a statistical technique for modelling time series and prediction. As described in background studies, section 2.1.2, ARIMA has three components, i) an autoregressive one that observes a growth or decline pattern in the time series, ii) an integrated one that observes the rate of change in growth or decline, and a iii) moving average that observes residual error between consecutive time points. ARIMA was selected in this study because it has been used for time series analysis by statisticians for several decades with promising results.

3.1.4. LSTM

Modern deep learning models are recently used in industries for different classification and regression tasks. For choosing a model correctly, in the first place, it was essential to define the objectives of the task and the properties of the input data. Since the dataset was a time series, the chosen approach should preserve the order of the data. For instance, convolutional neural networks do not support ordering and are mainly used for the matrix-like dataset. They are widely used for image processing and labelling while recurrent neural networks are currently used for temporal sequence processing.

RNNs have the problem of vanishing or exploding gradients. The models that are effectively used on real-world data for processing sequences are gated RNNs, including Long short-term memory (LSTM). LSTM is a supervised learning approach. Therefore, the input data should be reframed to make the problem a

semi-supervised learning task.

3.1.5. LSTM Multivariate

LSTM as a gated RNN supports different input structures as described in background studies 2.1.5. Therefore, it enabled us to train them with more information from neighbouring sensors. The input dataset for LSTM multivariate was created by forming a matrix of the average speed of successive nodes at each time step. Afterwards, a multivariate LSTM was implemented to learn the pattern from the upstream and downstream sensors to predict next time step average speed of a specific sensor in the selected geographic point.

3.1.6. Neural decomposition

A neural decomposition (ND) model was used as described in background studies in section 2.1.7. The weight initialisation was done for the network based on the original paper [8]. Besides, the scaling preprocessing technique was performed before training dataset to ND. The time stamp was scaled between 0 and 1, and the training data was scaled between 0 and 10 to avoid slow training for large numbers and local optima for small numbers.

3.2. The dataset

The dataset was gathered from the traffic detectors of Stockholm highways by the Swedish transport administration (Trafikverket) over 12 years. The sensors are located on all lanes approximately on every 300 meters of the highways. The detectors record the number of cars passing and the average speed every minute. Figure 10, shows the detectors in one situation.

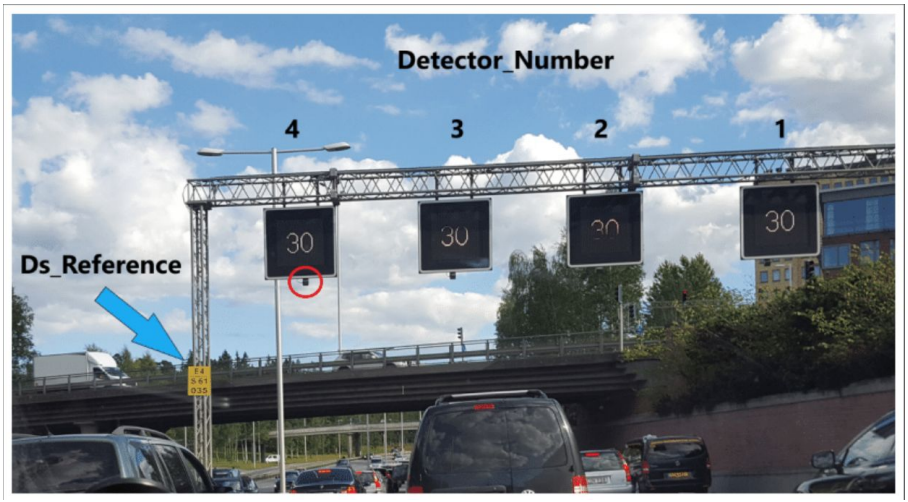


Figure 10: Traffic detectors situated on Stockholms' highways every 300 meters. Ds_reference contains the road name and the distance from the beginning of the road in meters. Detector_number shows the lane number, and they record the flow (number of passing cars) and the average speed every minute [11].

3.2.1. Structure of the MCS dataset

The file format structure is a comma-separated value (CSV), and it contains the following, table 1.

Variable	Description
Timestamp	Timestamp of the record (year, month, day, hour, minute)
Ds_Reference	Reference location (road name + distance from reference)
Detector_Number	Detector number (lane number)
Traffic_Direction	The direction of the reference location
Flow_In	Number of cars passed in one minute
Average_Speed	The average speed of the cars passed in one minute
Sign_Aid_Det_Comms	AID, Detector, MSI and Communication raw data
Status	Status of the detector
Legend_Group	Setup Group code of the detector
Legend_Sign	Sign type
Legend_SubSign	Frame type of sign
Protocol_Version	Protocol version of the detector Simone/TOP protocol

Table 1: Structure of MCS Dataset, variables in red are used in this thesis work.

Selected columns for this experiment are Timestamp, Ds-reference, detector number, average speed and status.

The time stamp is the date and time of the record. Ds_reference denotes where the detector is placed and contains the highway name, the direction, the distance of the location from the starting point in meters. Detector number shows the lane of the detector, flow in is the number of cars passed every minute, and the average speed is the average speed of passing cars. Status is a column to separate correct data from the wrong ones. Status '3', shows the healthy records that are used in this work. Also, there are some other signs of error codes, e.g. a speed greater than 250 km/h or flow rates greater than 250 vehicles/min are the signs of an error value. The density parameter is also computed for each record that is an essential parameter in traffic science and calculated as flow/speed.

3.2.2. Geographical location and selected sensor

Traffic sensors are located at approximately every 300 meters of the highways. Their names consist of the highway name, direction and the distance from a central location. A sensor in a traffic point was chosen to evaluate the prediction models. The selected traffic point should have both normal free flow traffic and congested status. It should be selected carefully so that the models can learn the different behaviour of the data. The sensor should have different traffic status during a day while it should not be too complex for any of the models. The Google maps tool is used to select this sensor.

Some sensors are shown in blue colour in figure 11. The selected sensor is shown in red colour. It is located on E4 highway south direction in Solna municipality toward downtown in Stockholm. As presented in figure 12, the roads live traffic status were in red during the rush hours on google maps, that means it was congested.

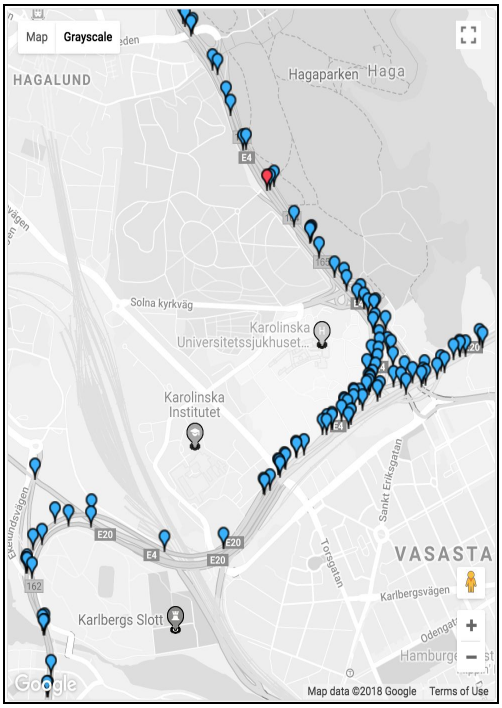


Figure 11. Some sensors are shown in the figure in blue colour. The chosen one is in red. The ds reference for the sensor is E4Z 60,270, ID:1189

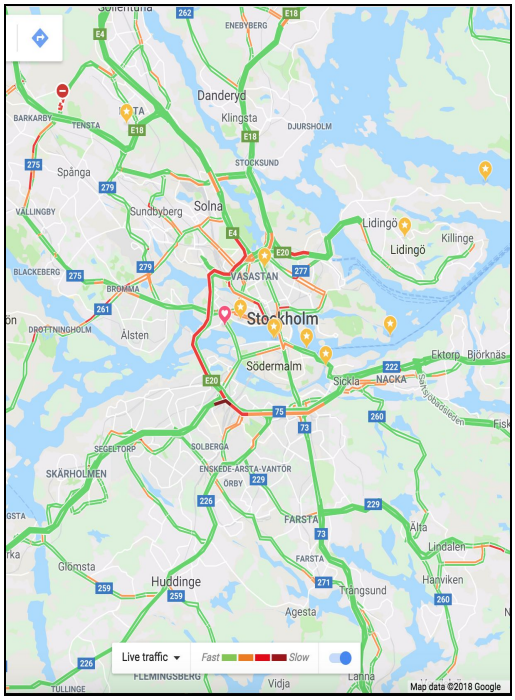


Figure 12. Stockholm traffic status, based on the Google map on a weekday 06:00 pm. Roads in red are congested.

3.2.3. The timeframe and forecasting horizons

The dataset contains 12 years of data from 2004 to 2016 from all the sensors. During these years many detectors have been added or removed from the highways. Each month of data is stored in a separated CSV file. In this experiment, the data from September 2016 was selected to study the results of the different prediction models. Forecasting horizons are determined to evaluate the prediction. Selected models were examined for short-term, mid-term and long-term prediction horizons. They were used to predict 1, 5 and 30 minutes in the future, and the whole test set, 9 days forward.

3.3. Data preparation

During the data preparation process, several techniques were used to prepare the dataset for training of prediction models. This phase was a very important one since the quality of input data has a substantial effect on the results of the task.

3.3.1. Data selection and cleaning

During data cleaning the error values from the dataset were removed, then the relevant columns were selected. The format of data was changed for instance, the timestamp format changed to a desired format. The name, direction and the distance of the records were derived from the Ds_reference column. The null values were filled with the last previous observed value. Alternatively, the mean value between the last and next observed values could be used.

3.3.2. Data transformation

Resampling, splitting, scaling, differencing, and time window techniques were used before training the predictors. The techniques are described in this section, and the usage for each model is explained in the result section 4.

3.3.2.1. Resampling

The data was time aggregated within the five-minute mean value of average speed to have another resolution of input data. In previous experiments on traffic problems [9-11], five-minute to fifteen-minute window time was selected for aggregating data. Five-minute aggregation was chosen because the traffic congestion does not last for a very long time in Stockholm usually. By using this technique, two input sets were created. The first one was the complete dataset that has the average speed every minutes and the second one was the time aggregated dataset that contains five-minute aggregated average speed. Both datasets were used to evaluate the prediction models. Resampling also determines the time steps for forecasting. Prediction for one time step in the future means next one minute on the complete dataset and next five minutes on the time aggregated dataset.

3.3.2.2. Splitting

A part of the data should be kept available to evaluate the prediction model. This is called the *test set*, and the reminder is called the *training set* that is used for training the model. Since the dataset is a time series preserving the order should be considered. The splitting was performed from the beginning of the dataset. First the length of the dataset was computed, then the length of training and test sets were computed based on the percentage. Then the first records until the training length were divided for the training task, and the rest was kept as the test set. For the whole experiments, the first 70% was used for training, and the rest 30% was used as the test set. In the LSTM models, 10% of the test set was also used for the validation test.

3.3.2.3. Scaling

Scaling is a technique commonly used in statistics and machine learning to squeeze the data between two values based on the usage. Scaling is a transformer, and after using, the results should be reverse scaled again to have realistic analysis. In this thesis work, before fitting the dataset to LSTM model, the average speed was scaled between 0 and 1 for the magnitude of the tanh activation function of LSTM cells. Furthermore, the timestamp was scaled between 0 and 1, and the average speed was scaled between 1 and 10, before training the Neural decomposition model. All the results were reverse scaled after prediction for computing errors.

3.3.2.4. Differencing

Differencing is a technique to make the data stationary. In a stationary time series, the mean and variance are not changing over time. This property makes prediction easier for mathematical models. Mathematical models find a transformation in a stationary time series between previously observed values and use the reverse transformation for prediction of future samples. Differencing removes trend, cyclic and seasonality components as described in background section 2.1.1. In this work, order one differencing was applied on the average speed value of dataset before training to the LSTM models. It computes the difference between

two successive observations. Similar to scaling, this transformation also needs to be reversed before evaluation tasks.

3.3.2.5. Time window

In supervised learning, there should be examples of paired input-output (labelled) data to let the model learn the correlation between input and label and perform the task of classification or regression. The dataset contains the traffic parameter value like the flow and speed of a traffic point per minute. Therefore, it was necessary to create a time window to get the next observed value as the label for the current parameter. This technique made the problem a semi-supervised learning task and enabled us to use deep learning model, specifically LSTM. For this work, the dataset was reframed to create a label for each value. The transformer function, gets the value of the next step as the label of the current step, for the whole dataset. It is also possible to use the transformer to reframe the dataset to different shapes. For example if for time step 't' the goal is to predict the value one step forward (t+1) it just needs to have the next value as the label and if the goal is to forecast 'n' steps in future, then it is possible to define (t+1, t+2,...,t+n) values as labels for the value of time t.

3.4. Evaluation criteria

In a regression prediction, the estimated values are quantitative. Therefore, it is meaningful to use the arithmetic operation to compute errors. In a well-fitting regression model, the predicted values are close to the actual data. There are several measures as presented in [2] to compute the distance between predicted values and the observed values. In this work, the root mean square error (RMSE) and the coefficient of determination (r^2) were used for evaluate the models. The time elapsed is computed for measuring the efficiency of the models. Moreover, two criteria, transition ratio and transition accuracy are introduced to evaluate the predictions for less frequent events.

3.4.1. Root Mean Square Error (RMSE)

One of the most commonly used evaluation criteria for a regression prediction problem is the Root mean square error (RMSE). RMSE measures the difference between actual values and the predicted values by Taking the square root of the average square errors. It is a straightforward measure to explain and penalise the more significant errors more than other measures like MEA. RMSE has an advantageous property, that is being in the same unit as the observed parameter. However, it should be noticed that RMSE is not necessarily increased with the variance of the errors [2]. In another word, if the data is uniform most of the time, a good result is achieved with this criterion while the predictor might not work convincingly. Here is the RMSE equation:

$$RMSE = \sqrt{1/n \sum_{j=1}^{n} (y_j - \hat{y}_j)^2} \tag{11}$$

3.4.2. The coefficient of determination (r^2)

Another well-known criterion for regression assessment as described in [2] is the Coefficient of determination (r^2). r^2 computes how much the regression explains test-data variation by taking the square of the sample correlation coefficient (i.e., r) between the observed values and the predicted values. In the equation (12), Sum of Squares Error (SSE) variance of error is the variation that is not explained by the regression, measures how far the data are from the predicted values. SSR denotes the variation that is explained by regression. Sum of Squares Total (SST) measures how far the data are from the mean value. The r2 equation is as follows:

$$r^2 = \frac{SSR}{SST} = \frac{SST - SSE}{SST} \tag{12}$$

For instance, r^2=0.8 means that the prediction regression model can explain 80% of the variation of the actual data.

3.4.3. Time elapsed

The running time should be computed for the models during training and prediction tasks to measure the cost of each model. Time elapsed is the time difference between the beginning of data transformation and the end of prediction for the whole test set. The time for reading data from the file system and the time for data cleaning and selection were excluded.

3.4.4. Transition Ratio

There should be a criterion to measure how the model can learn the transitions. Transitions are highly variable velocity changes. A transition is a radical change in the average speed between two consecutive time steps. In the traffic dataset, there are several spikes in changing the average speed. The spikes are very important to be predicted by the model. Gradual changes make the data uniform and cause small root mean square error. However, small RMSE does not necessarily indicate that the model is powerful enough to forecast extreme changes. The proposed criterion for a better evaluation is Transition ratio, that is the number of transitions predicted by the model in the prediction set over the number of transitions observed in actual test data.

A transition is a highly variable change in the input data from one time step to the next successive one, based on a threshold. To compute a meaningful threshold, we observed the dataset and figured out that there was a 30 Km/h difference between average speed in two consecutive time steps when the spike happened. The equation of the transition ratio is as follows:

$$
\begin{aligned}
&\textit{transitions in test set} = T = \left\{ \forall t \mid y_t - y_{(t-1)} > \textit{threshold} \right\} \\
&\textit{transitions in prediction set} = \widehat{T} = \left\{ \forall t \mid \widehat{y}_t - \widehat{y}_{(t-1)} > \textit{threshold} \right\} \\
&\textit{transition ratio} = \frac{|\widehat{T}|}{|T|}
\end{aligned}
\tag{13}
$$

3.4.5. Transition Accuracy

Another proposed measure is the transition accuracy that is defined as the proportion of the size of the intersection set and union set of two transition sets as stated before. The motivation for this measure was the Jaccard-similarity score. Since two sets may have different sizes, it is not possible to compute the actual Jaccard distance between them. Here is the equation of transition accuracy:

$$Transition\ Accuracy\ =\ \frac{|T \cap \widehat{T}|}{|T \cup \widehat{T}|} \tag{14}$$

3.5. Hyperparameter tuning and optimisation

There are several techniques to optimise an estimator. The forecast model could be run on the whole test dataset at once. Alternatively, the prediction can be performed on the test set in time intervals. Moreover, the hyperparameters that cannot be learnt by the model need to be tuned.

3.5.1. Walk forward method

In Time series prediction, sometimes the goal is to forecast multiple steps in the future. There are different approaches to optimise a model on a multi-steps prediction. In this experience, two methods were used.

1. A direct approach to predict the whole test set at once with the forecast models.
2. The second one was a rolling scenario known as 'Walk forward' method. Walk forward is an optimisation technique to find better parameters in a rolling approach. First, a time window is predicted, then the window is shifted forward for the next step. In this method, on each step, the model predicts one time window then the actual value is made available to the model to predict the next one. This scenario continues until the end of the test set.

3.5.2. Grid search

Grid search or parameter sweeping is a technique that is used for finding the optimal model in an estimator. In this technique, there is a range of values for every parameter. At each iteration, a combination of parameters is used to evaluate the model based on the criteria. The optimal model is the combination that results in the best values regarding all the evaluation measures. We chose to tune the hyperparameters manually by a grid search, to analyse all the selected evaluation criteria for each combination. Alternatively, we could use automatic approaches such as randomised search or other distribution searches such as uniform or exponential distribution search to have an optimal model in one of the evaluation criteria like MSE.

4. Results

In this chapter, the results of our experiments are presented on the selected models, persistent, the previous week observation, ARIMA, LSTM, LSTM multivariate and ND which are described in the methodology section 3.1. Afterwards, the models are compared concerning the previously stated criteria such as, the root mean square error (RMSE), the coefficient of determination (r^2), time elapsed, transition ratio and accuracy. Finally, the interpretation of the results are discussed.

4.1. Introduction to the results

Before evaluating the models, some understanding of the dataset is presented. The dataset as described in section 3.2, is the average speed of vehicles for every minute recorded by the selected sensor, E4Z 60,270, during the selected time frame, September 2016. It is called the original dataset in the rest of the experiments. The time aggregated dataset is the mean value of the average speed every five minutes of the original dataset as discussed in the resampling technique, section 3.3.2.1. The time steps in the original dataset are one minute and in the time aggregated dataset are five minutes. Each model was applied to the original dataset and the time aggregated dataset for the assessment task.

As stated previously in the splitting technique in section 3.3.2.2, the first 70% of the data was used for the training phase, and the remaining 30% used for the test. For each iteration the RMSE, r^2, time elapsed, transition ratio and accuracy were calculated. We were looking for lower RMSE and time elapsed and higher r^2 , transition ratio and accuracy.

In figures 15 to 38, the blue line is the actual average speed on the selected sensor at each time step, and the orange line is the predicted value. The blue dots are the transition points in the test set, and the orange dots are the transitions in the prediction set.

4.1.1. Dataset insights

A time series decomposition approach was used to analyse the additive components of the dataset. Additive components are components that can be used to reconstruct the original data.

As described in section 2.1.1, the trend, seasonal, and residual components of the datasets (average speed) were decomposed. Figure 13 shows the additive components of the original dataset and figure 14 shows the time aggregated dataset.

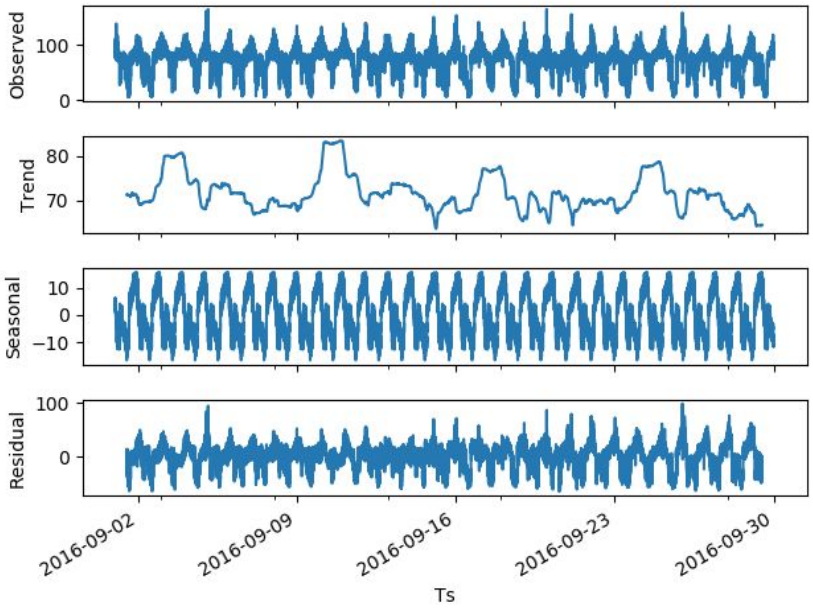


Figure 13: Additive components of the original dataset ‘speed’ (freq = 1 day)

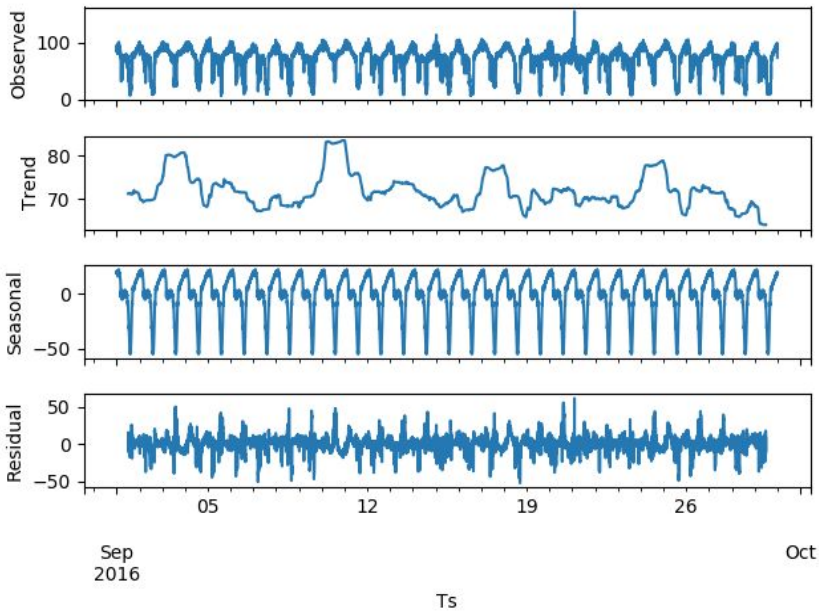


Figure 14: Additive components of the time aggregated dataset 'speed' (freq = 1 day)

As shown in the figures 13 and 14, the trend is not linear. There are a repetitive seasonality and the residual shows high variability in some time intervals. The trend in the original dataset and the time aggregated dataset are similar, and the seasonality component has a slightly smoother shape in the time aggregated data. The difference between the two datasets is mostly in the residual component. Time intervals that have high variance in the residual component are visible in both datasets. Therefore, the time aggregated dataset appears to be a proper representative for the original dataset.

4.1.2. A persistent model

The persistent model as mentioned previously in section 3.1.1, uses the *last observed value* as a prediction for the next step. This naïve model is chosen to be the baseline for the evaluation. First, the persistent model was applied to the original dataset, and the result is shown in figure 15. Then it was applied to the time aggregated dataset, and the result is shown in figure 16.

RMSE= 8.573
r^2 = 0.8731
Transitions ratio= 87/87
Transitions Accuracy= 17/157 = 0.108
Time elapsed= 0:00:00.014

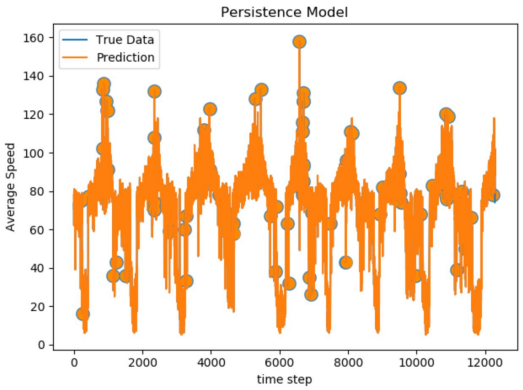


Figure 15: Persistent model on the original dataset

RMSE= 7.208
r^2 = 0.9037
Transitions ratio= 24/24
Transitions Accuracy=0
Time elapsed= 0:00:00.0031

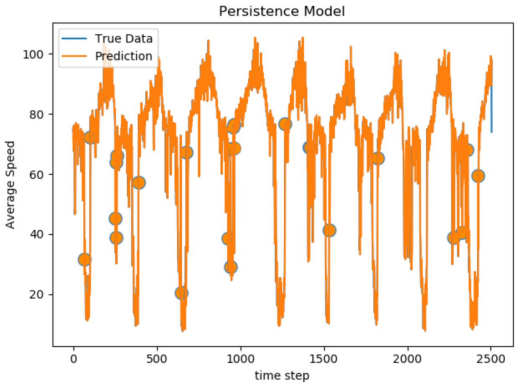


Figure 16: Persistent model on the time aggregated dataset

As shown in figure 20, The RMSE for the original dataset is 8.57 and r^2 is 0.87 which means that the regression model could explain 87% of the variation of the test dataset. These parameters on the time aggregated dataset are 7.20 and 90%.

Since the persistent model is the baseline for the evaluation, RMSE in the other models should not be higher than 8.57 on the original dataset, and 7.20 on the time aggregated dataset. Similarly, r^2 should not be less than 0.87 on the original dataset and 0.90 on the time aggregated dataset. The transition ratio in this model is 1. It means that all the transitions in the test dataset appeared in the prediction set which is an expected result because the model repeated previous value at each time interval. Therefore, all the transitions appeared in the prediction set one step later.

The persistent model was used for a 30 minutes forecast horizon to have a baseline for a multi-step forecast. It was 30 time steps forward for the original dataset, and the result is shown in figure 17, and for 6 time steps forward for the time aggregated dataset, the result is shown in figure 18.

	RMSE	r^2
t+1	8.562	0.8733
t+10	12.726	0.7203
t+20	14.096	0.6570
t+30	15.345	0.5939

Time elapsed 0:00:00.01

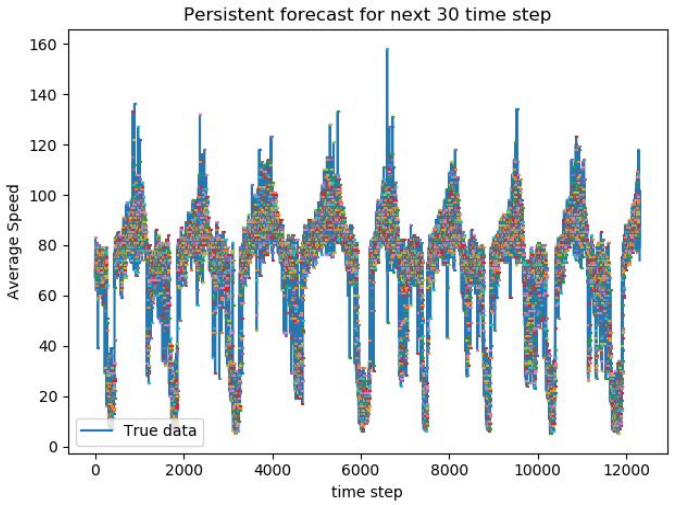


Figure 17: the persistent model for the multi-sequence forecast on the original dataset for the next 30 minutes (30 time steps)

	RMSE	r^2
t+1	7.204	0.9042
t+2	9.195	0.8440
t+3	10.086	0.8123
t+4	11.056	0.7744
t+5	12.001	0.7343
t+6	12.633	0.7056

Time elapsed 0:00:00.01

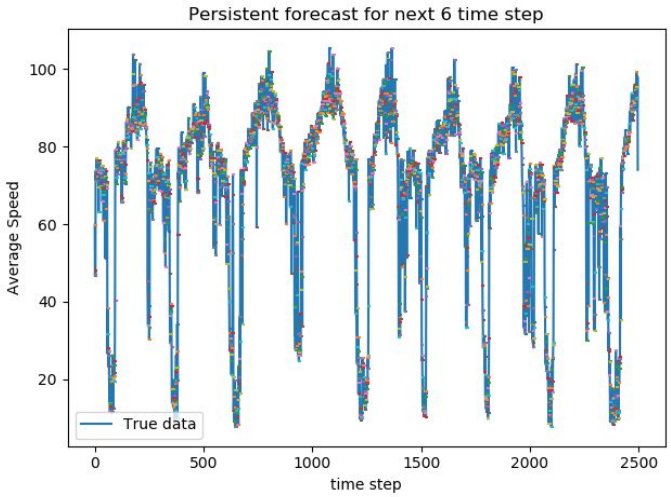


Figure 18: the persistent model for the multi-sequence forecast on the time aggregated dataset for the next 30 minutes (6 time steps)

Figure 17 shows that for a persistent model prediction on the original dataset, the RMSE at the first time step was 8.562 and it increased to 15.345 at the 30th step. Also, r^2 decreased from 87.33% to 59.39%. Figure 18 shows that in prediction on the time aggregated dataset, the RMSE at the first time step was 7.20 and increased to 12.63 in the 6th step. Moreover, r^2 reduced from 90.42% to 70.56%. This model is the baseline for our evaluation. Therefore, other models should give us better results, the RMSE should be less than the baseline value, and r^2 should be higher than the baseline value.

The persistent model was also studied for predicting the whole test set for long-term forecast horizon.

RMSE= 24.097
r^2 = -0.0022
Transitions ratio= 0/87
Transitions Accuracy= 0
Time elapsed= 0:00:00.01

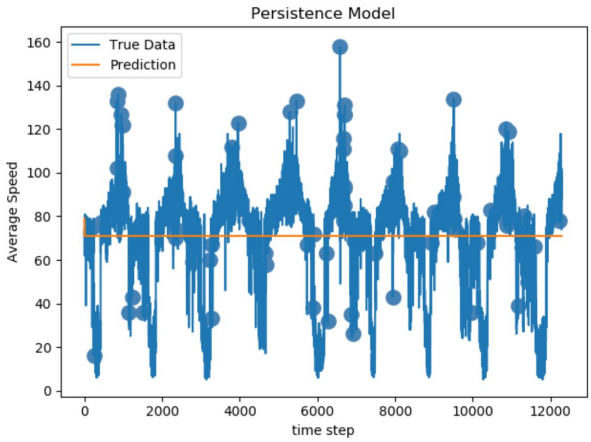


Figure 19: Persistent model with the direct approach on the original dataset.

RMSE= 23.606
r^2 = -0.0327
Transitions ratio= 0/24
Transitions Accuracy=0
Time elapsed= 0:00:00.003

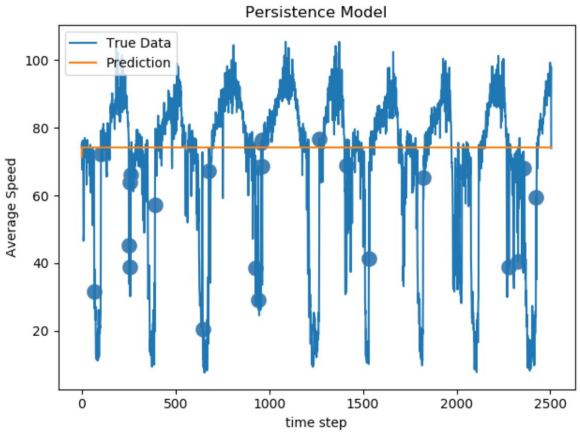


Figure 20: Persistent model with the direct approach on the time aggregated dataset

As presented in figure 19 and 20, if the persistent model is applied to predict the whole test set at once, the result is a straight line of the last observed value.

4.1.3. The previous week observation model

The previous week observation as described in section 3.1.2, is another naïve model. In this model the predicted value, is the observed value at the same day and time in the previous week. First, the model was applied to the original dataset and the result is shown in figure 21. Then it was applied to the time aggregated dataset, with the result being shown in figure 22.

RMSE= 37.370
r^2 = -1.4104
Transition ratio= 121/87
Transition Accuracy= 0.010
Time elapsed= 0:00:00.0144

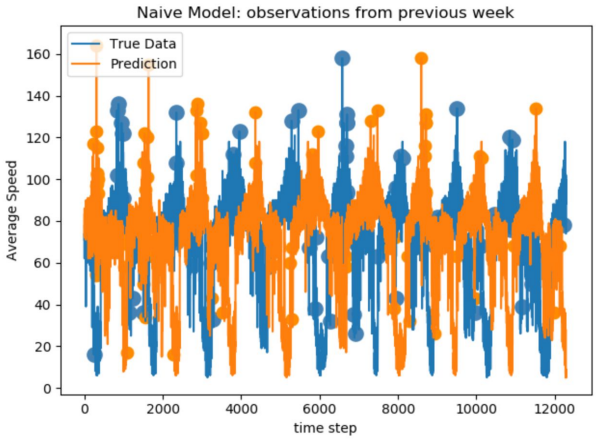


Figure 21: Previous week observation model on the original dataset

RMSE= 12.599
r^2 = 0.7058
Transition ratio= 27/24
Transition Accuracy= 0
Time elapsed= 0:00:00.0029

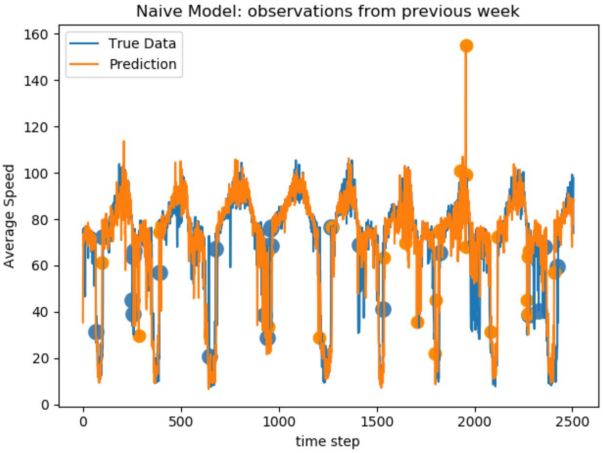


Figure 22: Previous week observation model on the time aggregated dataset

The RMSE and r^2 values in figures 21 and 22 show that the previous week model results are much weaker than the persistent model. The reason is that the previous week observation is very dependant on the dataset. If two successive weeks had similar patterns, this model could provide an acceptable result. The RMSE on the original dataset is 37, that is a very high value for an error measure for the average speed.

The RMSE on the time aggregated dataset is also 12.59 which is again much higher than the persistent model. r^2 for the dataset is negative which means the model does not follow the dataset trend. For the time aggregated dataset, r^2 is improved that means in the 5-minute time aggregated dataset, the variation of the data could be explained better within the same model. The transition ratio shows the proportion of the number of spikes in the observed week to the previous week.

4.1.4. ARIMA

ARIMA, as described in section 3.1.3, is an autoregressive integrated moving average model. Three parameters should be regularised to have an optimised model, the number of autoregressive lags and moving average lags and the order of difference. As explained in section 3.5.2, a grid search technique was used to find the best combination of the parameters (p: lag value=1, d:difference order=1, q: moving average=0). Also a walk forward technique, as described in section 3.5.1, was used to predict the whole test set. First, the model was applied to the original dataset, and the result is shown in figure 23. Then it was applied to the time aggregated dataset, and the result is shown in figure 24.

RMSE= 8.093
r^2 = 0.8870
Transition ratio=4/87
Transition Accuracy= 4/87=0.046
Time elapsed= 0:28:25.45

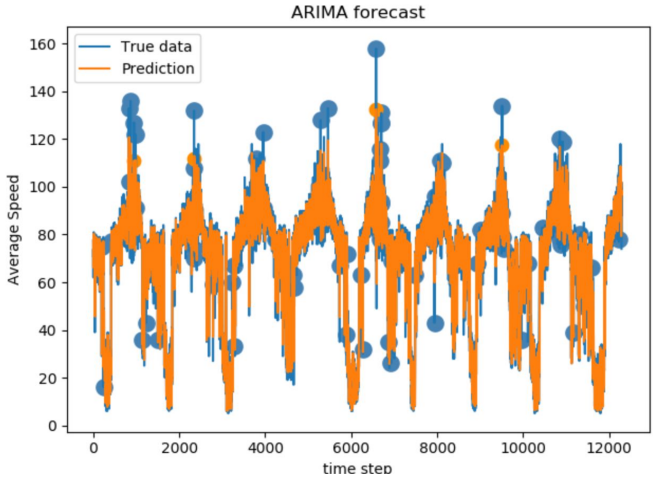


Figure 23: ARIMA model on the original dataset

RMSE= 7.087
r^2 = 0.9069
Transition ratio= 5/24
Transition Accuracy= 0
Time elapsed= 0:01:31.09

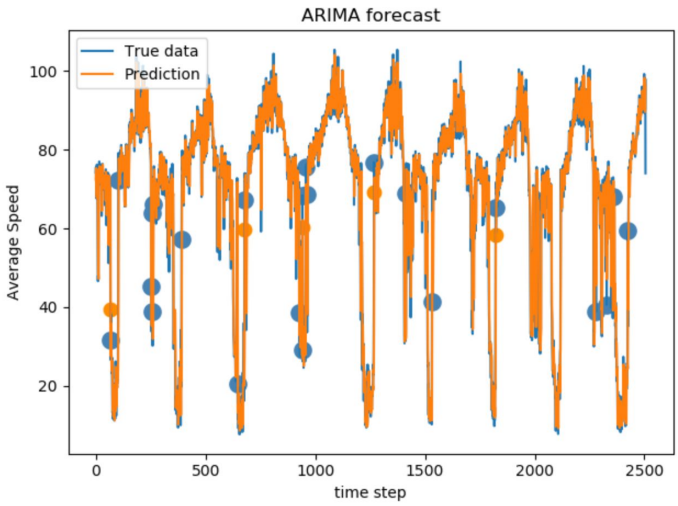


Figure 24: ARIMA model on the time aggregated dataset

As shown in figure 23 and 24, the RMSE on the original dataset is 8.09 which is less than the baseline value of 8.75 and on the time aggregated dataset is 7.08 which is less than 7.20. r^2 shows that ARIMA can explain 88.70% of data variation on the original dataset and 90.69% of the data variation on the time aggregated dataset. Both of them are higher than the r^2 in the baseline, 87.31% and 90.37% respectively. On the original dataset 4 transitions out of 87 were predicted, and on the time aggregated dataset 5 out of 24 were recognised.

ARIMA was also studied for predicting the whole test set. In this approach, the whole test set was predicted at once. Parameters were found by a grid search (p:lag value=1, d:difference order=1, q:moving average model=0). First, the model was applied to the original dataset, and the result is shown in figure 25. Then it was applied to the time aggregated dataset, and the result is shown in figure 26.

RMSE= 24.836
r^2 = -0.0646
Transition ratio= 0/87
Transition Accuracy= 0
Time elapsed= 0:00:00.36

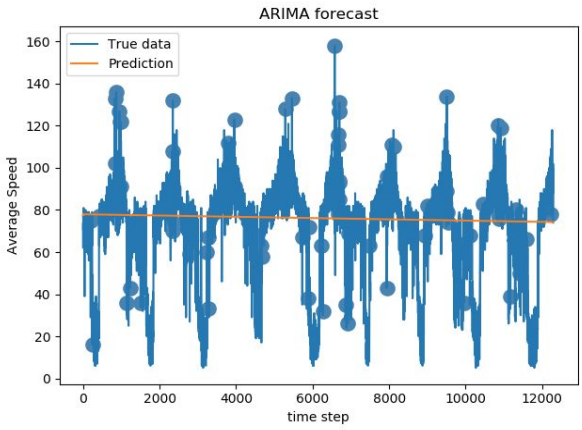


Figure 25: ARIMA model with the direct approach on the original dataset.

RMSE= 23.290
r^2 = -0.0052
Transition ratio= 0/24
Transition Accuracy= 0
Time elapsed= 0:00:00.17

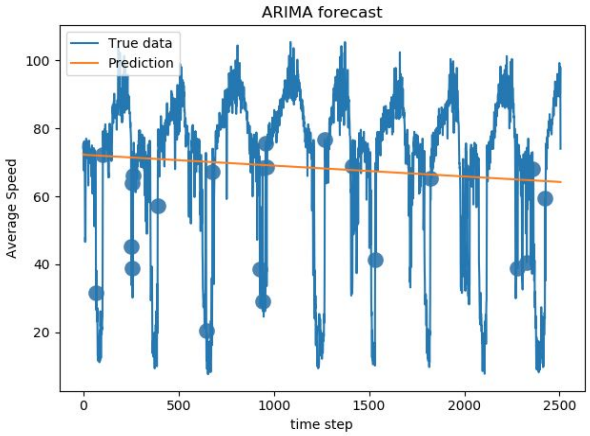


Figure 26: ARIMA model with the direct approach on the time aggregated dataset

As presented in figure 25 and 26, if ARIMA was applied to predict the whole test set at once, it results in a straight line.

ARIMA was used for the multi-step forecast of 30-minute time horizon using walk forward technique. 30-minute time horizon means thirty time steps forward for the original dataset, and the result is shown in figure 27, and six time steps forward for the time aggregated dataset, and the result is shown in figures 28.

	RMSE	r^2
t+1	8.0876	0.8871
t+10	12.205	0.7431
t+20	13.655	0.6787
t+30	14.920	0.6166

Time elapsed 0:33:06.72

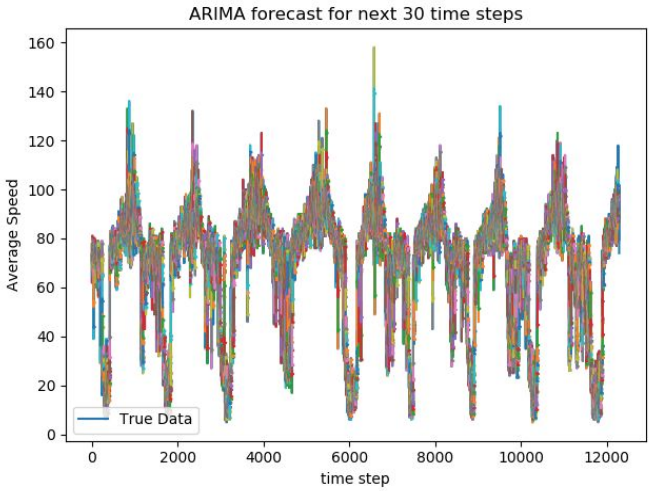


Figure 27: ARIMA for multi-sequence forecast using walk forward technique on the original dataset for the next 30 minutes (30 time steps)

	RMSE	r^2
t+1	7.0783	0.9072
t+2	8.9718	0.8510
t+3	9.9169	0.8180
t+4	10.921	0.7793
t+5	11.829	0.7411
t+6	12.484	0.7118

Time elapsed 0:01:44.68

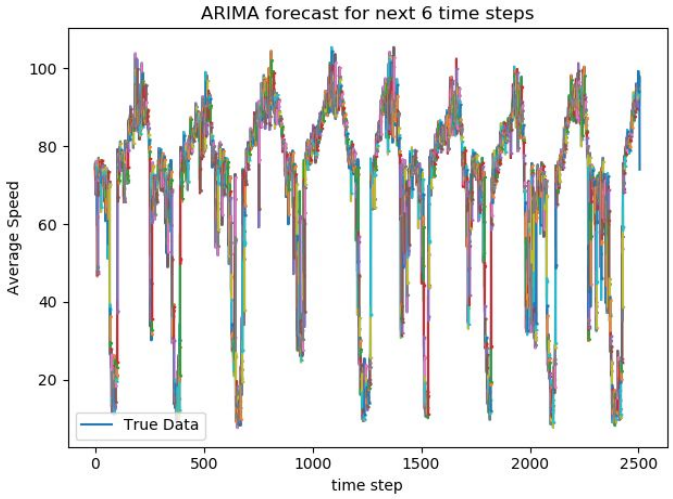


Figure 28: ARIMA for forecast using walk forward technique on the time aggregated dataset for the next 30 minutes (6 time steps)

Figure 27 shows that in ARIMA model multi-sequence forecast on the original dataset, the RMSE in the 30th step is 14.920 which is less than baseline value

15.345 and r^2 is 61.66% which is higher than 59.39%. In figure 28, the results on the time aggregated dataset show that the RMSE in the 6th step was 12.48 which is less than the baseline 12.63 and r^2 is 71.18% that was higher than 70.56%.

4.1.5. LSTM

Before using LSTM, the differencing technique was used to make the data stationary. Then it was changed to semi-supervised learning, by using time window technique and finally, the dataset was scaled between -1 and 1. All the techniques are described in Data transformation 3.3.2.

Many experiments were performed using a grid search technique as described in section 3.5. to find an optimum model in LSTM. Experiments show that, by adding layers more number of transitions are predicted and by adding neurons in each layer, the time elapsed gets shorter. Increasing epochs leads to finding more transitions while raising the time elapsed.

Based on the results, the optimum model in LSTM had five hidden layers of 10 LSTM neurons and one dense layer for the output. The selected loss function was MSE and Adam optimiser was used for optimisation. The outcome of using LSTM on the original dataset is presented in figure 29, and the result of using on the time aggregated dataset is presented in Figures 30.

RMSE = 8.196
r^2 = 0.884
Transition ratio = 5/87
Transition Accuracy = 0.01
Time elapsed = 0:29:30.82

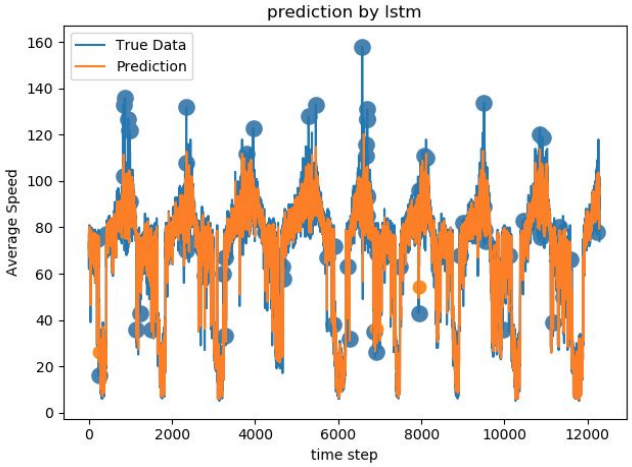


Figure 29: LSTM model on the original dataset.

RMSE = 7.1
r^2 = 0.9089
Transition ratio = 15/24
Transition Accuracy = 0
Time elapsed = 0:06:56.17

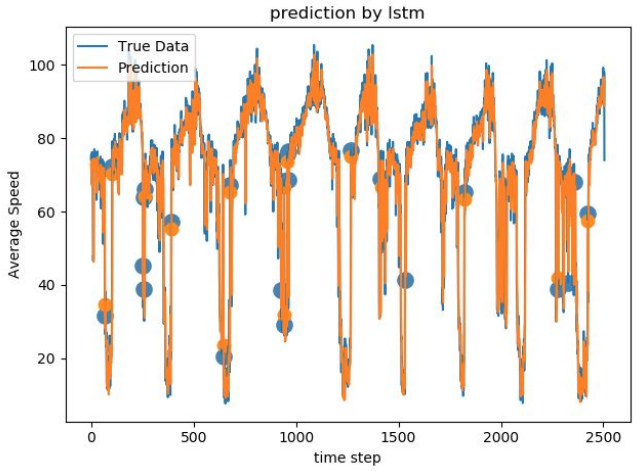
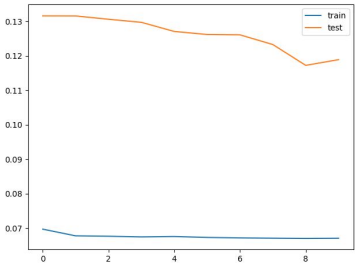


Figure 30: LSTM model on the time aggregated dataset.

As presented in figures 29 and 30, RMSE on the original dataset is 8.19 which is less than the baseline value 8.75, and RMSE on the time aggregated dataset is 7.1 that is less than the baseline value 7.20. r^2 shows that LSTM can explain 88.40% of data variation on the original dataset and 90.89% of data variation on the time aggregated dataset. Both of them are higher than baseline r^2, 87.31% and 90.37% respectively. On the original dataset, 5 transitions out of 87 were predicted, while on the time aggregated dataset 15 out of 24 was recognised. Note, *none* of the transitions could be predicted on the exact time they happened.

LSTM was also used to predict the whole test set on the original dataset, and the results are shown in figure 31, the time aggregated dataset, and the result is shown in figure 32.

RMSE = 12.025
r^2 = 0.7504
Transition ratio = 27/87
Transition Accuracy = 0.01
Time elapsed = 0:37:44.76



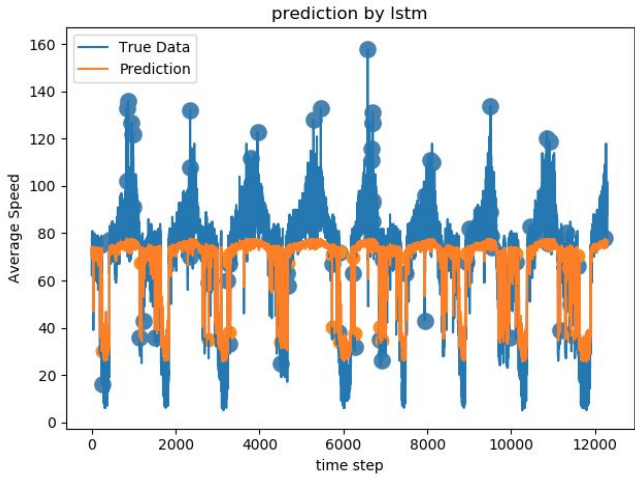
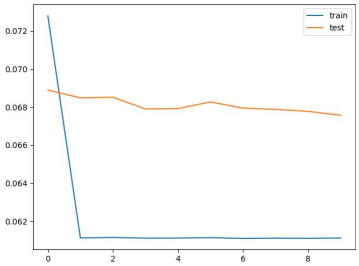


Figure 31: LSTM and cross-validation with the direct approach on the original dataset.

RMSE = 8.219
r^2 = 0.8748
Transition ratio = 26/24
Transition Accuracy = 0.02
Time elapsed = 0:06:02.42



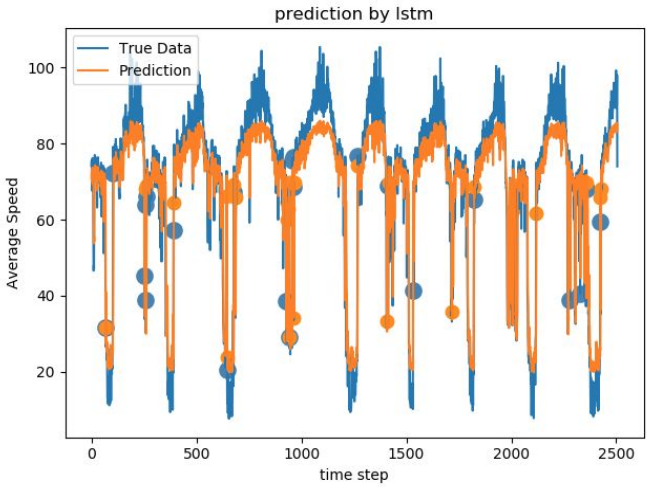


Figure 32: LSTM and cross-validation with the direct approach on the time aggregated dataset.

As presented in figures 31 and 32, the results is not as good as LSTM using the walk forward technique from a RMSE and r^2 perspective. However, it could learn 27 transitions out of 87 on the full test set and 26 transitions out of 24 on the time

aggregated test set. Two of the transitions could be predicted on the exact time they happened.

LSTM was used for the multi-step forecast of 30-minute time horizon using walk forward technique. 30-minute time horizon means thirty time steps forward for the original dataset, and the result is shown in figure 33, and six time steps forward for the time aggregated dataset, and the result is shown in figures 34.

	RMSE	r^2
t+1	8.1547	0.8850
t+10	12.135	0.7456
t+20	13.721	0.6750
t+30	15.133	0.6050

Time elapsed 0:26:58.231

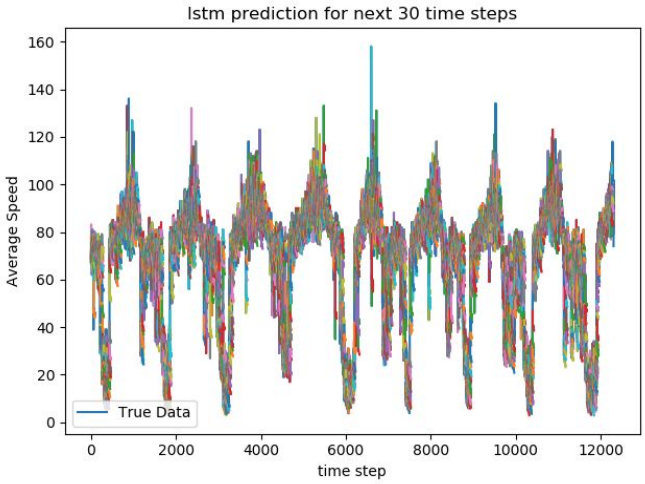


Figure 33: LSTM for multi-sequence forecast using walk forward technique on the original dataset for the next 30 minutes (30 time steps).

	RMSE	r^2
t+1	6.9931	0.9093
t+2	8.8842	0.8537
t+3	9.8689	0.8195
t+4	10.892	0.7802
t+5	11.816	0.7415
t+6	12.522	0.7096

Time elapsed 0:06:56.21

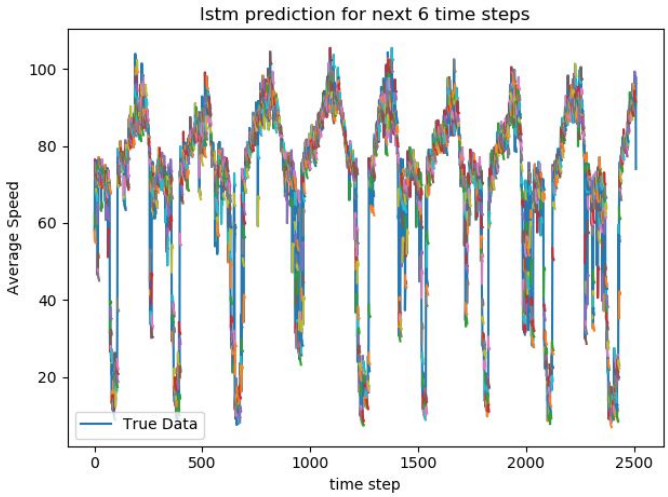


Figure 34: LSTM for multi-sequence forecast using walk forward technique on the time aggregated dataset for the next 30 minutes (6 time steps).

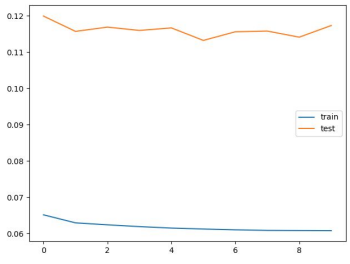
In figure 33 it is shown that in LSTM model for the multi-sequence forecast, the RMSE in the 30th step, is 15.133 which is less than baseline value 15.345 and r^2 is 0.6050 which is higher than 59.39%. Figure 34 shows that the RMSE in the 6th step is 12.522 which is less than the baseline 12.63 and r^2 is 70.96% that is higher than 70.56%.

4.1.6. LSTM Multivariate

LSTM was studied with more neighbourhood information from upstream and downstream. The input was the history of the average speed for the sensor, one successor and one predecessor sensor. A differencing technique was used to make the data stationary before training the model.

Then it was changed to a semi-supervised learning, by using time window technique and finally, the dataset was scaled to (-1, 1). All the techniques are described in section 3.3.2. First, the model was applied to the original dataset, and the result is shown in figure 35. Then it was applied to the time aggregated dataset, and the outcome is shown in figure 36.

RMSE = 11.691
r^2 = 0.7644
Transition ratio = 4/87
Transition Accuracy =0.01
Time elapsed = 0:37:22.34



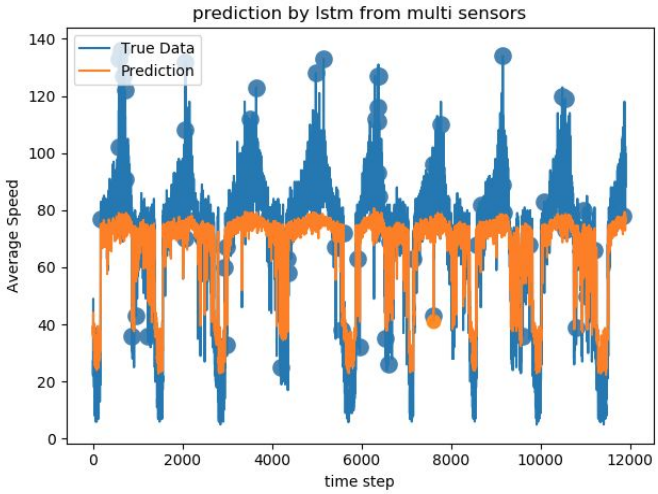
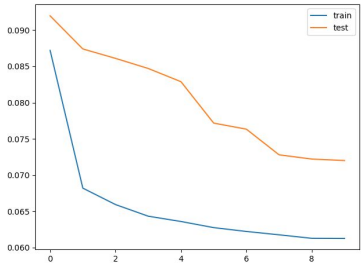


Figure 35: LSTM multivariate model and the cross-validation on the original dataset.

RMSE = 7.807
r^2 = 0.8871
Transition ratio = 21/24
Transition Accuracy = 0
Time elapsed = 0:07:34



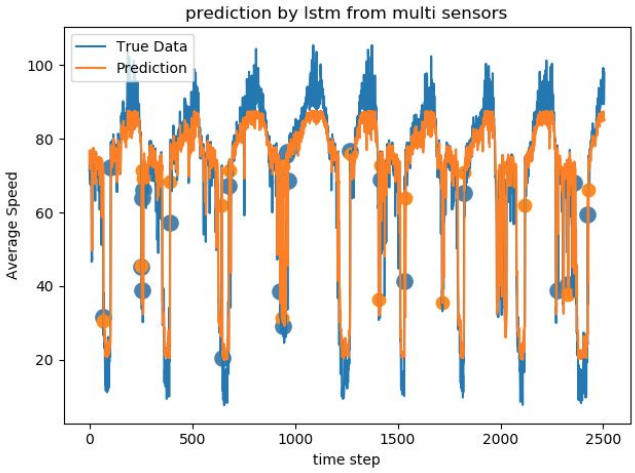


Figure 36: LSTM multivariate model prediction and cross-validation on the time aggregated dataset.

As presented in figures 35 and 36, RMSE on the original dataset is 11.691 which is higher than the baseline value 8.75 and RMSE on the time aggregated dataset is 7.80 again higher than 7.20. r^2 shows that LSTM multivariate can explain 76.44% of data variation on the original dataset and 88.71% of data variations on the time aggregated dataset. Both of them are below the baseline r^2, 87.31% and 90.37% respectively. On the original dataset 4 out of 87 are predicted, on the time aggregated dataset 21 out of 24 are recognised.

4.1.7. Neural decomposition (ND)

Neural decomposition as described in section 3.1.6 was applied to the original dataset and the time aggregated dataset. Before running ND on the dataset, the time step t was scaled between 0 and 1, and the average speed was scaled between 1 and 10 to keep the model away from slow training for large numbers and also preventing local optima for small numbers. A grid search technique was used to sharpen the results concerning the stated criteria in this model. The network had ten neurons, and 100 epochs were used for training on the original dataset, and the result is shown in figures 37. For the time aggregated dataset, The network had ten neurons, and 400 epochs were used for training, and the result is shown in figures 38.

RMSE = 28.932
r^2 = -0.4447
Transition ratio = 0/87
Transition Accuracy = 0
Time elapsed = 0:10:11

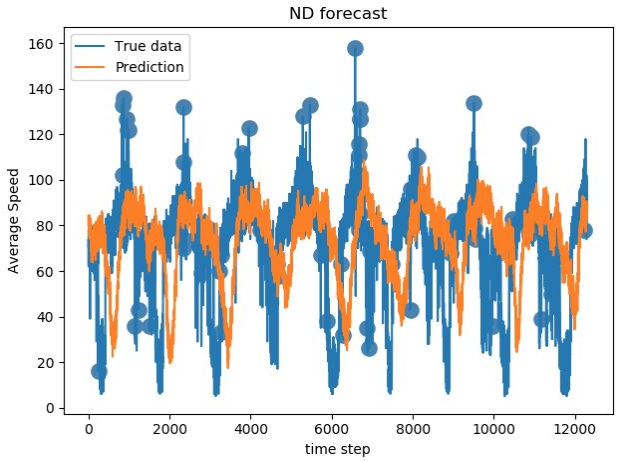
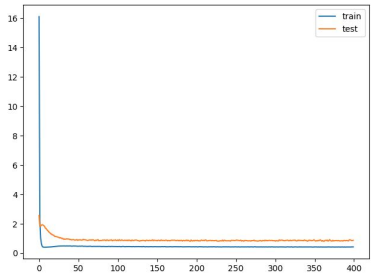


Figure 37: Neural decomposition model on the original dataset

RMSE = 12.525
r^2 = 0.7093
Transition ratio = 0/24
Transition Accuracy = 0
Time elapsed = 0:02:24



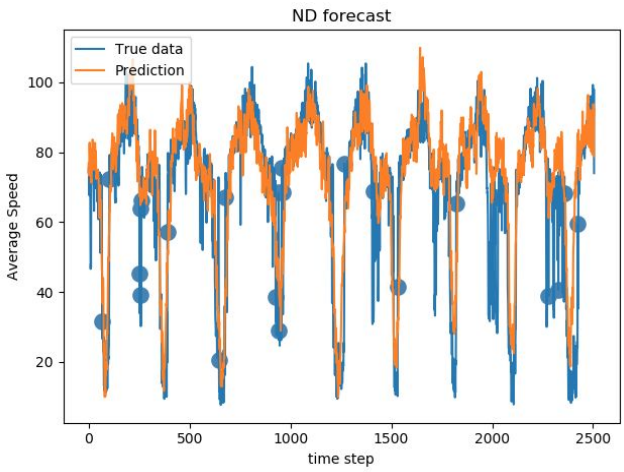


Figure 38: Neural decomposition model on the time aggregated dataset

As presented in figures 37 and 38, the neural decomposition model could learn the nonlinear repetitive pattern of the dataset. RMSE on the original dataset is 28.93 which is a very high error value for the average speed forecast and r^2 for the original dataset is a negative value that shows that ND was not successful to explain the variation of data on the original dataset.

For the time aggregated dataset, the RMSE on the time aggregated dataset is 12.52, and r^2 shows that the ND could explain 70.93% of data variation. None of the transitions in the test set is predicted by the neural decomposition model.

4.2. Comparison

In this section, all the results are projected in a table format to facilitate the evaluation task by comparing models side by side. In table 2, the selected models are compared for long-term horizon forecast on the original dataset. Table 3 shows the results for short-term horizon forecast on the original dataset. Table 4 contains the result of mid-term horizon forecast on the original dataset. Measures in red are the best results at each table and values in bold are the baseline measures.

model	RMSE	r^2	transition ratio	transition acc	time elapsed
Persistent	**24.097**	**-0.0022**	**0/87**	**0**	0:00:00.01
ARIMA	24.836	-0.064	0/87	0	0:00:00.36
LSTM	12.025	0.7504	**27/87**	0.01	0:37:44.76
LSTM- multivariate	**11.691**	**0.7644**	4/87	0.01	0:37:22
Neural decomposition	28.932	-0.4447	0/87	0	0:10:11.36

Table 2: Comparison of the prediction for long-term horizon on the original dataset

model	RMSE	r^2	transition ratio	transition acc	time elapsed
Persistent	**8.573**	**0.8731**	**87/87**	**0.1**	0:00:00.014
Previous week obs.	37.370	-1.4104	121/87	0.01	0:00:00.0144
ARIMA	**8.093**	**0.8870**	4/87	0.046	0:28:25.45
LSTM	8.196	0.8840	**5/87**	0.01	0:29:30.82

Table 3: Comparison of the prediction for short-term horizon on the original dataset

model	RMSE (t+1)	r^2 (t+1)	RMSE(t+30)	r^2 (t+30)	time elapsed
Persistent ARIMA LSTM	8.562 8.0876 8.1547	0.8733 0.8871 0.8850	**15.345** **14.920** 15.133	**0.5939** **0.6166** 0.6050	0:00:00.01 0:33:06.72 0:26:58.23

Table 4: Comparison of the prediction for mid-term horizon on the original dataset

In table 5, the selected models are compared for long-term horizon forecast on the time aggregated dataset. Table 6 shows the results for short-term horizon forecast on the time aggregated dataset. Table 7 contains the result of mid-term horizon forecast on the time aggregated dataset. Measures in red are the best results at each table and values in bold are the baseline measures.

model	RMSE	r^2	transition ratio	transition acc	time elapsed
Persistent ARIMA LSTM LSTM- multivariate Neural decomposition	**23.606** 23.290 8.219 **7.807** 12.525	**-0.0327** -0.005 0.8748 **0.8871** 0.7093	**0/24** 0/24 **26/24** 21/24 0/24	**0** 0 **0.02** 0 0	0:00:00.003 0:00:00.17 0:06:02.42 0:07:34 0:02:24

Table 5: Comparison of the prediction for long-term horizon on the time aggregated dataset

model	RMSE	r^2	transition ratio	transition acc	time elapsed
Persistent Previous week obs. ARIMA LSTM	**7.208** 12.599 **7.087** 7.1	**0.9037** 0.7058 0.9069 **0.9089**	**24/24** 27/24 5/24 **15/24**	**0** 0 0 0	0:00:00.0031 0:00:00.0029 0:01:31.09 0:06:56.17

Table 6: Comparison of the prediction for short-term horizon on the time aggregated dataset

model	RMSE (t+1)	r^2 (t+1)	RMSE (t+6)	r^2 (t+6)	time elapsed
Persistent	7.204	0.9042	**12.633**	**0.7056**	0:00:00.01
ARIMA	7.078	0.9072	**12.484**	**0.7118**	0:01:44.68
LSTM	6.993	0.9093	12.522	0.7096	0:06:56.21

Table 7: Comparison of the prediction for mid-term horizon on the time aggregated dataset

5. Discussion

Experiments show that ARIMA results on the selected sensor data was unsatisfactory based on the transition accuracy alone. ARIMA results was so close to the persistent model that was chosen as the baseline based on RMSE and r^2. ARIMA could predict a few number of highly variable velocity changes and none of them were predicted at the exact time they happened.

LSTM performance was not acceptable on the information from the local area. The outcomes were similar to the naive model based on RMSE and r^2 criteria. LSTM could predict only half of the transitions on the time aggregated dataset and they were not predicted accurately.

Neural decomposition was not able to provide satisfactory results on the data from the selected sensor. It was observed that the Neural decomposition was *very* dependant on the input data. There was a noticeable difference between the results on the original dataset and on the time aggregated dataset.

None of these models could provide a noticeable improvement from the persistent model in short term and mid term forecast horizons. There was at most 5% improvement based on RMSE and 3.8% based on r^2 compare to the persistent model. Additionally, the transition ratio was sometimes worse than the persistent model.

All the stated models had a similar performance on the time aggregated dataset based on RMSE and r^2 with less computation time. Similarly, higher number of transitions could be predicted on the time aggregated dataset. Therefore, the resolution of the dataset affects the regression prediction task.

6. Limitations

One of the problems in prediction is where we cannot predict early enough. Realistically, the predictions may not be fast enough for real-time traffic control. This is really a limitation of the setting. Trafikverket data contains measurements every 300m, and are collated every minute. In our work, we looked where speeds transitions occur and two examples are given in figure 39. Actually, congestion on resolutions smaller than the measurements is the inference problem [16].

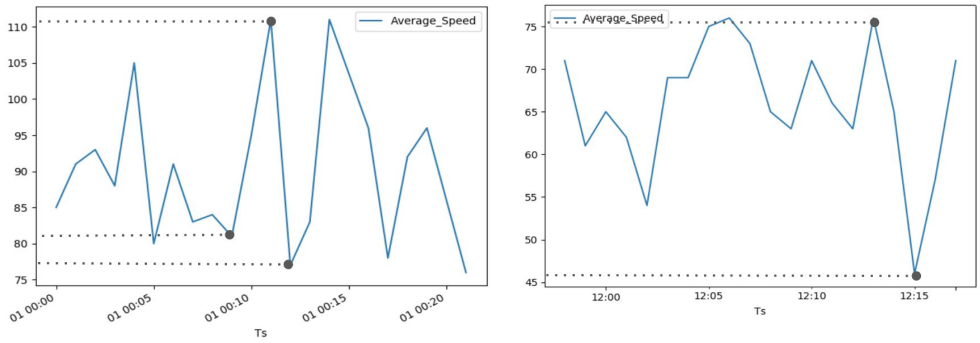


Figure 39: Two examples of transition threshold in the dataset 30 km/hour.

Figure 39 shows the rationale for a transition threshold of 30 km/hour. However, an improvement for future work would be to use the mean+std of the average speed. Alternatively, Zainab Abbas et al. in [11] looked at the density as a congestion indicator, with the disadvantage that, density needs to be calculated whereas average speed is directly available from the sensor.

Besides, the average speed is a traffic parameter that has information over space and time. Therefore, it is more informative than flow or density. Additionally, The average speed can be used directly for travel planning by the drivers and driving assistance systems.

Evaluation measures are critical factors for any assessment task. The root mean square error (RMSE) and the mean relative error (MRE) are the commonly used

for evaluating prediction models as studied in [9-12]. Using RMSE and r^2 is not enough to decide whether the model is suitable for a regression prediction. Because these measures aggregate the error values for the whole test set. Since most of the time the data is changing smoothly, aggregation of the error eliminates the extreme changes. Radical increase or decrease in the average speed might be less frequent but critical for a dataset like traffic parameters. Therefore it is essential to have a metric that can assess the ability of the models for forecasting rare events.

Another issue for predicting traffic parameters is the effect of lanes. During the data wrangling phase, we observed that the pattern in lane one was entirely different from lane two. The reason was the entrance and exits to the first lane. Lane two and three had relatively similar behaviours, but lane four had again slightly different pattern due to the high speed. Also merging and splitting of the roads increased the complexity of the dataset. We focused on lane two, to decrease the effect of lanes. Alternatively, we could study more neighbouring sensors on the other lanes.

Another limitation was that the selected sensors information were not enough to provide superior results by selected models compared with the naïve persistent model. Though we can not conclude if adding more sensors would improve the results of the models. Besides, highly variable velocity changes might happen because of the driver's behaviour and without any signals from historical data thus none of the models could provide acceptable results based on transition accuracy.

7. Recommendations for Trafikverket

1. We needed a measurement metric that is acceptable for evaluating prediction models. Trafikverket, a road authority, needs a *domain specific measure*. Whereas ML models use measures like root mean squared error (RMSE) and r^2 , we propose a **new** metric the transition ratio, which is when the speed **changes** by a given threshold. Essentially it means rapid slow downs and speed ups, but of course we are most interested in slow downs, as they lead to congestion.

2. We needed a larger partition of the traffic network to create a usable prediction model. More number of sensors should be selected to forecast highly variable changes in the average speed of passing vehicles accurately. The optimum size of the partition can be investigated in another study.

3. Time aggregation can provide a smoothed value of the speed. Clearly, learning the very high resolution changes is not what we are after. From looking at our data, 5 minutes was an acceptable resolution over which to smooth the data, thereby permitting learning, but not at a too low resolution to lose the flow characteristics.

8. Conclusions

The following four questions were posed:

Q1. Is it possible to predict the average speed at a geographical point independently, accurately and efficiently? Do added upstream and downstream points improve the prediction?

Q2. Does the model perform sufficiently in predicting highly variable velocity changes prediction?

Q3. Is the model able to outperform traditional methods for different forecasting horizons? Are there any improvements based on the selected criteria, namely RMSE, r^2, transition ratio and accuracy?

Q4. How does the time aggregation of the input data affect the prediction results? Does it have a positive or negative effect? By time aggregation we mean taking averages from the original 1 minute samples (which are already time-averaged) to 5 minutes.

ARIMA, LSTM and Neural Decomposition models were implemented to answer these questions. The models were evaluated to predict the average speed of a specific traffic point for three forecasting horizons. A naïve model namely persistent model was implemented as a baseline for evaluation. Proposed criteria were used to measure the highly variable velocity changes prediction. The multivariate LSTM model was implemented to add upstream and downstream knowledge to the model. The models were applied to two input datasets, the original dataset and the time aggregated dataset that was 5-minute aggregation of the original dataset.

A1. It was expected that the selected models could predict the average speed at a spatiotemporal point with the nearby points knowledge independently, accurately and efficiently. ARIMA and LSTM had a performance just 5% better than the baseline and needed a longer computational time. None of the models could

provide superior results than the persistent model for short term and medium term forecasting horizons. This could have happened because there was not enough information in the selected local area of the sensor and more number of sensors is needed to provide a local prediction.

A2. Predicting highly variable velocity changes was hard for the selected models, with few number of sensors. Only a few number of transitions could be predicted by ARIMA. Half of the transitions could be predicted by LSTM on the the time aggregated dataset. None of the models could predict the transitions at the exact times they happened.

A3. None of the models provide acceptable results based on the transition accuracy, since there were not enough signals in the selected neighbourhood area. The RMSE and r^2 results were close to the baseline for short term and midterm time horizons. A larger geographical area should be studied to compare the models. Thresholds for the transitions could also be explored

A4. We can conclude that the time aggregation on the dataset did not have a negative effect on the prediction task. All the models in this experiment had a better performance on the time aggregated dataset. They could provide lower RMSE and higher r^2 with less computation time. However, the input information was insufficient to compare the models based on the transition accuracy.

9. Lessons learned

Data visualisation helped us to gain a deep insight into the dataset. We had a better understanding of the traffic parameters such as flow, speed and density when we visualised some subsets of the original dataset. The subsets were selected from different months, roads and lanes. Data visualisation enabled us to find a part of the dataset with fewer error states and missing values for the analysis. Also, we investigated how the velocity changed over different lanes of a road.

The quality of the dataset and preprocessing techniques had a direct impact on the forecast results. For instance, solutions to fill empty or invalid values were substantial to provide an error free input stream. The Transformation of data before training the models was an essential task in this study, namely, scaling and time window techniques.

Commonly used measures are essential for the assessment. However, some features might be hidden if only one measure is used. Different criteria should be used to be able to measure different features that were not detected by other measures. RMSE was the measure we used at the beginning of the experiments to evaluate the models and we added transition ratio and accuracy for measuring highly variable changes. But none of the models could provide acceptable results regarding transition accuracy.

Besides, the goal of the prediction should be determined during the first steps to be able to find a suitable model. For example, the forecast horizons and predicting highly variable velocity changes were essential in our work. In another word, it is better to start from the question not from the data.

More sensors should be selected as an input to the prediction models. Slowdowns happen as a result of driver's behaviour and probably there were no signals of highly variable changes in velocity in the selected neighbouring sensors. Probably sensors located farther away could provide some information about the highly changes in average speed.

10. Future work

The input data should be changed to cover a larger geographic area. Adding more neighbouring sensors from further area can provide more information about highly variable velocity changes. Upstream sensors with longer distance can have signals of the passing vehicles that will pass the sensor in the next time intervals. Downstream sensors can have signals of slow moving queue that will be extended to the sensor in the coming time steps. Besides, the effective distance that improves the prediction of highly variable changes can be investigated.

More features can be added to the input vector before training. For instance, day time or night time, weekdays and weekends. These features can provide more information for the models. The behaviour of the models can be studied on the added features.

From our analysis [19] the statistical properties of each lane differ. The autocorrelation of the slower, or inside, lane is much higher than that of the faster, or outside, lane. Given that the ACF and the partial ACF[2] differ, there is some case for using a model for each lane. That said, road merges and splits would be more difficult in this time of approach.

[2] previous autocorrelations are removed in a partial ACF

Glossary

Aggregated: Combine the measures based on a function (mean-value)

Direct: Predict the whole test at once, we used this technique to evaluate the models for long-term forecast horizon.

Time aggregated: Compute the mean value over a time interval (five-minute)

Transition: A highly variable velocity change based on a threshold.

Transition accuracy: measures if the transition is predicted in the exact time step that happened in the actual data.

Transition ratio: measures the ratio of predicted transitions over actual transitions.

Walk forward method: Predict the test set in a rolling scenario, we used this technique to evaluate the models for mid-term forecast horizon.

References

[1] I. Goodfellow, Y.Bengio, A. Courville, (2017). *Deep learning*. Cambridge: MIT Press, pp.363-476.

[2] R. Jain, (1991). *The art of computer system performance analysis: techniques for experimental design, measurement, simulation, and modelling*. New York: John Wiley and Sons, Inc, pp.221-229.

[3] R.J. Hyndman, G. Athanasopoulos, (2018). *Forecasting principles and practice*. [Online]. Available: https://otexts.org/fpp2/ . [Accessed: October 8, 2018]

[4] R. Adhikari, R. K. Agrawal, (2013). *An Introductory Study on Time Series Modeling and Forecasting*. [Online]. Available: https://arxiv.org/pdf/1302.6613.pdf . [Accessed: May 20, 2018]

[5] C. Olah, (2015). Understanding LSTM Networks. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs . [Accessed: October 17, 2018]

[6] A. Karpathy, (2015). The Unreasonable Effectiveness of Recurrent Neural Networks [Online]. Available: http://karpathy.github.io/2015/05/21/rnn-effectiveness . [Accessed: October 17, 2018]

[7] K. Greff, R. k. Srivastava, J. Koutnik, B. R. Steunebrink, J. Schmidhuber, (2015). LSTM: A Search Space Odyssey [Online]. Available: https://arxiv.org/abs/1503.04069 . [Accessed: October 20, 2018]

[8] L.B. Godfrey, M.S. Gashler, (2017). Neural Decomposition of Time-Series Data for Effective Generalization [Online]. Available: https://ieeexplore.ieee.org/document/7955052 . [Accessed: October 20, 2018]

[9] M. Fouladgar, M. Parchami, R. Elmasri, A. Ghaderi, (2017). Scalable Deep Traffic Flow Neural Networks for Urban Traffic Congestion Prediction [Online]. Available: https://arxiv.org/abs/1703.01006 . [Accessed: October 20, 2018]

[10] Th. Epelbaum, F. Gamboa, J. M. Loubes, J. Martin, (2017). Deep Learning applied to Road Traffic Speed Forecasting [Online]. Available: https://arxiv.org/abs/1710.08266 . [Accessed: October 20, 2018]

[11] Z. Abbas, A. Al-Shishtawy, S. Girdzijauskas and V. Vlassov, "Short-Term Traffic Prediction Using Long Short-Term Memory Neural Networks," 2018 IEEE International Congress on Big Data (BigData Congress), San Francisco, CA, 2018, pp. 57-65. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8457731&isnumber=8457596 . [Accessed: November 16, 2018]

[12] N. A. Consuegra Rengifo, 'Detection and Classification of Anomalies in Road Traffic using Spark Streaming', Dissertation, 2018. [Online]. Available: http://kth.diva-portal.org/smash/record.jsf?pid=diva2:1261957

[13] T. Thorri Sigurdsson, 'Road traffic congestion detection and tracking with Spark Streaming analytics', Dissertation, 2018.

[14] J. Reginbald Ivarsson, 'Scalable System-wide Traffic Flow Predictions Using a Partitioned Transportation System and Recurrent Neural Networks', Dissertation, 2018.

[15] A. Håkansson, "Portal of Research Methods and Methodologies for Research Projects and Degree Projects," in The 2013 World Congress in Computer Science, Computer Engineering, and Applied Computing WORLDCOMP 2013; Las Vegas, Nevada, USA, 22-25 July. CSREA Press USA, 2013, pp. 67–73. [Online]. Available: http://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A677684&dswid=5912

[16] J. Brownlee, (2018), 'Getting Started with Applied Machine Learning' https://machinelearningmastery.com/start-here/#timeseries

[17] Y. Lv, Y. Duan, W. Kang, Z. Li and F. Wang, "Traffic Flow Prediction With Big Data: A Deep Learning Approach," in IEEE Transactions on Intelligent

Transportation Systems, vol. 16, no. 2, pp. 865-873, April 2015. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6894591&isnumber=7070850 [Accessed: October 20, 2018]

[18] Y. Duan, Y. Lv and F. Wang, 'Performance evaluation of the deep learning approach for traffic flow prediction at different times', 2016 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), Beijing, 2016, pp. 223-227. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7551691&isnumber=7551648 [Accessed: October 20, 2018]

[19] I. Marsh, B.Bjurling, 'End of queue detection in road traffic', RISE SICS AB. [online]. Available: http://bada.sics.se/wp-content/uploads/2017/12/eoq2018.pdf [Accessed: November 16, 2018]

[20] H. P. Sajjad, K. Danniswara, A. Al-Shishtawy and V. Vlassov, 'SpanEdge: Towards Unifying Stream Processing over Central and Near-the-Edge Data Centers', *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, Washington, DC, 2016, pp. 168-178. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7774704&isnumber=7774340 [Accessed: November 16, 2018]

[21] N. Laptev, S. Smyl, S. Shanmugam 'Engineering Extreme Event Forecasting at Uber with Recurrent Neural Networks', 2017 [Online]. Available: https://eng.uber.com/neural-networks/ [Accessed: November 26, 2018]

[22] A. Katachova, 2013, Time Series ARIMA Models [online]. Available: https://www.youtube.com/watch?v=Y2khrpVo6qI&t=1581s [Accessed: December 3, 2018]

[23] The Swedish Transport Administration - Trafikverket, 'Annual Report 2017', 2018 [Online]. Available: https://trafikverket.ineko.se/Files/sv-SE/49148/Ineko.Product.RelatedFiles/2018_086_TRV_Annual%20Report_2017.pdf [Accessed: December 3, 2018]

Appendix A (Codes, Environment and Tools)

The code of the thesis work is available on the following Address:
https://github.com/Cosarg/Deeplearning-on-traffic-data-

The following tools are used in this thesis work:
Python 2.7 an interpreted high-level programming language.
https://www.tensorflow.org/

PyCharm an integrated development environment.
https://www.jetbrains.com/pycharm/

Jupyter a web-based notebook for interactive development environment.
https://jupyter.org/

Apache Spark a framework for large-scale data processing.
https://spark.apache.org/

Scikit-Learn a python library for machine learning and statistical computations.
https://scikit-learn.org/

TensorFlow a library for high performance numerical computation with support for machine learning and deep learning.
https://www.tensorflow.org/

Keras a high-level library on top of Tensorflow that support design and implementation of neural networks.
https://keras.io/

HopsWorks a platform that integrates state of the art services for Data Science, Data Engineering and Machine Learning tasks.
https://www.hops.io/

TRITA -EECS-EX-2019:786