Comparative Analysis of LSTM, ARIMA, and Facebook's Prophet for Traffic Forecasting: Advancements, Challenges, and Limitations

Ziyar Uzel

Supervisor(s): Elena Congeduti

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Ziyar Uzel
Final project course: CSE3000 Research Project
Thesis committee: Elena Congeduti, Georgios Iosifidis

Abstract

Accurate short-term traffic forecasting plays a crucial role in Intelligent Transportation Systems for effective traffic management and planning. In this study, the performances of three popular forecasting models are explored: Long Short-Term Memory (LSTM), Autoregressive Integrated Moving Average (ARIMA), and Facebook's Prophet, for short-term traffic prediction. The models were trained and evaluated using a dataset of traffic flow data collected from 161 detectors over a specific time period. The experimental results reveal that ARIMA outperformed LSTM and Prophet in terms of Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). This suggests that while deep learning methods, such as LSTM, are generally acknowledged to outperform ARIMA in short-term traffic forecasting, this study reveals that there are specific scenarios where such well-accepted fact needs to be tested.

1 Introduction

In the current state of the modern world, the number of vehicles on the roads in large cities has reached a great extent that the roads and their infrastructure are unable to hold. To reduce the burden on the existing transportation infrastructure, two different approaches can be considered: extending the current infrastructure and utilizing traffic control strategies. The former is not only extremely expensive but also not viable in some cities [4]. Whereas the latter approach requires minimal expenditure and can help to exploit the existing infrastructure in a more efficient way by offering better travel decisions to drivers and reducing traffic congestion [9]. These traffic control strategies together with increasing data-collecting infrastructure in the cities compose what is known as Intelligent Transportation System (ITS) [8]. To leverage the advantages of ITS, accurate and reliable traffic information and data must be available. This emphasizes the critical role of accurate short-term traffic predictions in supporting ITS operations. [10].

2 Background and Related Work

Short-term traffic prediction differs from conventional traffic forecasting methods due to its focus on a shorter time horizon, typically ranging from a few minutes up to around 45 minutes [4, 11]. By way of formal definition, let x_t represent the vehicle count at a specific detector at discrete time t. short-term traffic forecasting that is referred to throughout this paper can be formulated as follows:

$$x_t = f(x_{t-1}, x_{t-2}, \ldots, x_{t-k}),$$
$$k = 1, 2, 3, \ldots$$

In the formulation provided, the variable k represents the number of previous time steps taken into account for predicting the current traffic count x_t. It determines the length of the historical data window used by the forecasting model. This formulation captures the dependence of the current vehicle count on the previous counts, allowing for a one-step forecast.

With the rapid increase in the availability of real-time traffic data, many models are put to use to solve the problem of accurate short-term traffic forecasting [4, 8]. These prediction models can be divided into three categories: naive methods, parametric models, and non-parametric models [10].

Naive methods are methods that do not rely on any model assumption. and they are fast and easy to implement due to low computational effort. These methods include using historical averages and Instantaneous Travel Times (ITTs). The accuracy of these methods is low [10].

Parametric models are the approaches where "the structure of the model is predetermined" [10]. In other words, the model's structure, such as the functional form and parameters, is specified in advance based on assumptions or prior knowledge. Autoregressive integrated moving average (ARIMA), also called a Box-Jenkins model, is one of the most popular statistical and parametric techniques that is utilized for traffic forecasting. ARIMA combines autoregressive (AR), integrated (I), and moving average (MA) components. The AR component considers the dependency on previous observations, the I component deals with differencing to achieve stationarity, and the MA component captures residual errors from past observations. ARIMA is effective in capturing temporal dependencies, trends, and noise in the data, making it suitable for traffic forecasting. Ahmed and Cook conducted one of the earliest research (in 1979) on the performance of ARIMA, where they compared ARIMA with other ad hoc smoothing methods, and concluded ARIMA's superiority in accuracy [16]. During the same era, Levin and Tsao had put a research comparing different configurations of ARIMA to find out the most statistically significant model in traffic forecasting [15]. Kalman filtering is another parametric approach used in traffic forecasting that models the system as a set of linear equations with Gaussian noise, and it estimates the system state by recursively updating its estimate based on measurements from the system [17].

There are also recent parametric approaches that are developed for time-series forecasting. Facebook's Prophet is one of those approaches. In 2017, Facebook's data science team published a paper titled "Forecasting at Scale", in which they introduced a new model called Prophet. It combines trend modeling, seasonality decomposition, and holiday effects to capture underlying patterns in the data. Prophet utilizes a Bayesian framework, providing uncertainty estimates for accurate and interpretable forecasts [27]. In a 2022 study by ChikkaKrishna et al., the performance of Prophet, along with another Facebook forecasting model, was evaluated in the domain of short-term traffic forecasting in India. The authors suggested comparing these models with other forecasting methods like ARIMA and LSTM to gain a better understanding of the Prophet's effectiveness in the mentioned domain [28].

Non-parametric models consist of methods that are flexible in the quantity of their parameters. Such models' structure and parameters are decided after the observation of the data. Some of the non-parametric approaches use neural-networks

[4, 19], k-nn [20], support vector machine (SVM) [18], and support vector regression (SVR) [21].

Being the central model of the paper and a non-parametric method, Long Short-Term Memory (LSTM) is a special type of recurrent neural network (RNN) that is extensively utilized for analyzing and predicting sequential data [7]. Unlike conventional feedforward neural networks, LSTM networks use memory cells that have the ability to retain and recall information over extended temporal contexts [8]. This unique attribute empowers LSTM networks to effectively capture long-term dependencies and discern patterns within time-series data [8]. It consists of multiple layers of memory cells, and each cell contains input, output, and forget gates, which controls the flow of information. Through these gates, the network can selectively retain or discard information based on its relevance to the current task. LSTM has gained significant importance across diverse domains, including natural language processing, speech recognition, and time-series forecasting[7,8]. There are a vast amount of papers available that evaluate the performance of LSTM in short-term traffic forecasting, some of them include [4,7,8].

3 Knowledge Gaps and Research Question

Short-term traffic forecasting is a widely studied domain that has seen the proposal of numerous models [2, 4, 5, 6, 7, 10, 12, 13, 20, 28]. However, despite the advances, there exist significant knowledge gaps that hinder the effectiveness of these models.

One such gap is the inadequate consideration of external factors, like weather and events, in most existing models, as emphasized by Wang and Zhang [23]. These factors play a crucial role in influencing traffic flow, yet their impact is not fully incorporated into the forecasting process. Another neglected aspect is the potential improvement in prediction accuracy through the inclusion of spatial correlation between different locations [24]. While the significance of this correlation has been demonstrated, it is not consistently accounted for in the modeling process.

Additionally, the challenge of handling missing data in traffic datasets remains unresolved [22]. Missing data poses a significant obstacle in accurately predicting traffic patterns, and developing effective techniques to address this challenge is imperative for reliable forecasting. Moreover, the robustness of traffic forecasting models has not been thoroughly examined across various situations. To ensure the effectiveness of these models, it is crucial to evaluate their performance in different scenarios and develop adaptive methods accordingly [36].

Furthermore, the field of traffic forecasting offers a diverse set of techniques that can be utilized to tackle this complex problem, and moreover, the continuous advancements in research and innovation contribute to the emergence of new models. Even though deep learning based methods, such as LSTM, have been shown to outperform traditional statistical models, it is still not clear if this is the case for every situation and context given the high variability in traffic patterns across different scenarios. In addition, a consistent comparison of more novel models as Facebook's Prophet with state-of-art methods for traffic forecasting is essential to evaluate their effectiveness and identify potential limitations [1,3,4,6,25,26,34,37].

Addressing these knowledge gaps in short-term traffic forecasting can significantly contribute to the development of more accurate models. This paper specifically focuses on the last gap by comparing various forecasting models, namely LSTM, ARIMA, and Facebook's Prophet, in Den Haag, Netherlands, aiming to provide valuable insights to short-term traffic forecasting. This will be achieved through answering the **research question**: *"Does LSTM outperform ARIMA and Prophet in the domain of short-term traffic forecasting?"*

4 Methodology

To conduct the comparison analysis, all three models, namely LSTM, ARIMA, and Prophet, were implemented. This section outlines the methodology adopted for each model, including the data pre-processing, model configuration, and evaluation metrics used. For a better understanding of the methodology followed in the implementation of the models, a brief explanation of the data and evaluation metrics are necessary. So in section 3.1 a description of the data is provided, in section 3.2 an explanation for the selected evaluation metrics is present and in the following sections, 3.3-3.4, the models' methodology is explained.

4.1 Data

The data that is used throughout the models is collected from Den Haag Municipality Dataset. The data corresponds to 15-minute vehicle counts from inductive-loop traffic detectors collected during the month of November 2019. These detectors are located in the lines of President Kennedylaan and Johan de Wittlaan. They span 11 intersections and add up to 162 detectors in total. In essence, each detector is an array of size 2880 (# of 15 minutes in a month) and each entry of the dataset corresponds to a 15-minute vehicle count.

As for the handling of the missing data, due to the importance of ensuring the integrality of the dataset, the missing or invalid data entries are replaced by the previous data entries in temporal order instead of getting dropped [4].

4.2 Evaluation Metrics

The evaluation metrics play a crucial role in the comparison analysis of LSTM, ARIMA, and Prophet models. In this study, the root mean squared error (RMSE) is chosen as the primary metric for both training and evaluation. Additionally, the mean absolute percentage error (MAPE) is used as a complementary metric to measure and evaluate the performance of the models.

RMSE is a widely used evaluation metric in the domain of time series forecasting [4, 6, 7, 8, 11]. It provides a measure of the average prediction error, taking into account the differences between the predicted and actual values see the formula below:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2}.$$

where x_i is the actual value, $\hat{x}_i$ is the predicted value, and n is the number of observations. The use of RMSE allows for direct comparison with previous researches in the field.

To complement the evaluation with a relative error metric, MAPE is chosen. MAPE provides insights into the accuracy of the models by expressing the prediction error as a percentage of the actual value, see the following formula:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_i - \hat{x}_i}{x_i}\right| \times 100\%$$

However, one limitation of MAPE and other relative error metrics is the potential for division by zero. In scenarios where the actual value is zero, dividing by zero can lead to numerical instability and large values.

To address this issue, after making the predictions, all the zero values with their corresponding predictions are dropped and neglected in the computation of the performance errors. By removing these zero values, the problem of division by zero is mitigated, ensuring more stable and meaningful evaluation results.

By utilizing both RMSE and MAPE, a comprehensive evaluation of the models' performance can be obtained, capturing both the absolute and relative errors in the predictions.

4.3 LSTM

The LSTM model is essentially a type of RNN. It is a chain of memory cells and consecutive data points are fed into these cells sequentially to predict the very next data point. Figure 1 demonstrates such a structure.

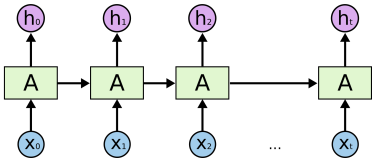

Figure 1: A represents a memory cell, x_t represents the input values, and h_t represents the output value

The computation of LSTM takes place within each of these memory cells, as depicted in Figure 2.

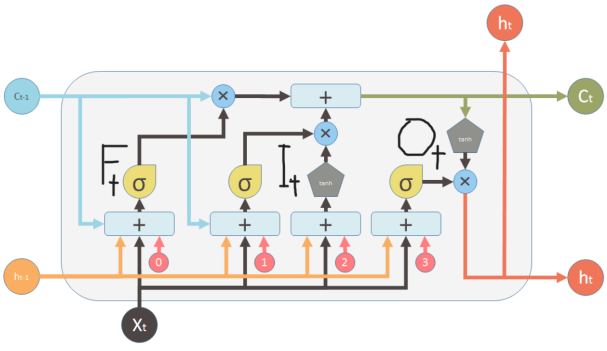

Figure 2: C_t represents cell memory at time step t, σ is the sigmoid activation function, pentagon represents $\tanh$ activation function, red circles numbered from 0-3 represent biases.

As mentioned before in Section 2, memory cells consist of three gates: forget (F_t in the illustration), input (I_t), and output (O_t) gates, see Figure 2. Output of these gates then can be used to obtain the memory cell value C_t and output value h_t. Formulation for all of these operation is as follows:

$$F_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_0)$$
$$I_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_1)$$
$$O_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_3)$$
$$C_t = F_t \cdot C_{t-1} + i_t \cdot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_2)$$
$$h_t = O_t \cdot \tanh(C_t)$$

In these equations, the forget gate F_t determines the amount of previous cell state to forget. The input gate I_t decides how much of the new information should be stored in the cell state. The output gate O_t controls the amount of cell state to reveal as the output. The cell state C_t represents the memory at time step t, and the output h_t represents the prediction at time step t. The sigmoid function σ and the hyperbolic tangent function $\tanh$ are activation functions used to transform the inputs. The weights W_{xi}, W_{hi}, W_{xf}, W_{hf}, W_{xo}, W_{ho}, W_{xc}, W_{hc}, and biases b_0, b_1, b_2, b_3 are adjustable parameters of the LSTM model that are learned during the training process.

1. **Data Pre-processing**: Data pre-processing for LSTM model mainly involved normalization, sliding window technique, and splitting data into training and test set.

   Due to their common usage in the domain, Z-score and min-max normalization are considered for the normalization function [11, 30]. Both of their performance are evaluated by running the LSTM model on the training set and comparing the RMSE. Min-max normalization performed better out of the two, see Figure 3 and 4, hence selected to be the normalization function.

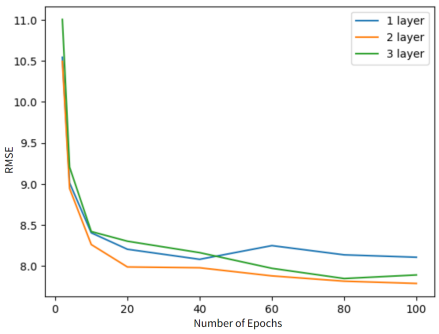

Figure 3: Min-max normalization performance.

Moreover, to feed into the LSTM model, each detector column, representing a univariate sequence, was transformed into multiple samples. Each sample consists of a specified number of time steps [29]. This data transformation technique is referred to as the "sliding window" approach throughout the paper. To help illustrate the concept of the sliding window, consider the following univariate sequence:

x_{t-4}	x_{t-3}	x_{t-2}	x_{t-1}	x_t

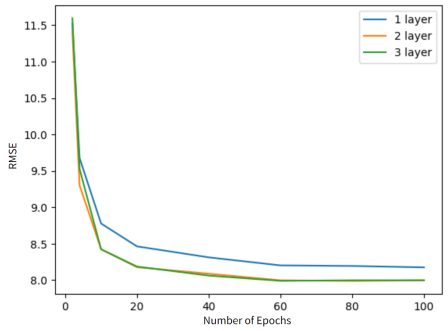

Figure 4: Z-score normalization performance

By applying the sliding window technique with a window size of 3, the sequence is transformed into a matrix as shown below:

x_{t-4}	x_{t-3}	x_{t-2}
x_{t-3}	x_{t-2}	x_{t-1}
x_{t-2}	x_{t-1}	x_t

Now, instead of feeding a single data entry from the initial sequence, each row of this matrix can be fed as a single unit into the LSTM model. The LSTM model will then make predictions for the next time interval. For example, after the first row is fed, the LSTM model will predict x_{t-1}. This allows for capturing the temporal dependencies in the data and making accurate predictions.

Moreover, the data is divided into a training set containing 80% of the data and a test set comprising the remaining 20%. This same split is applied to the other two models, namely ARIMA and Prophet.

2. **Model Configuration**: The LSTM model requires the specification of various parameters, including the size of the sliding window, the number of layers, the choice of loss function, and the activation functions.

   After conducting thorough experimentation and analyzing the results, it was observed that a sliding window size of 80 yielded the best performance in terms of lower RMSE. Hence, a sliding window size of 80 was adopted for the LSTM model.

   Additionally, the performance comparison between layer 2 and layer 3 revealed insignificant differences, see Figure 3. To maintain model simplicity and avoid unnecessary complexity, layer 2 was selected as the preferred option.

   Regarding the choice of loss and activation functions, default/common functions were utilized. The mean squared error was employed as the loss function to quantify the model's prediction accuracy. For the activation functions, the sigmoid function was applied to the input, forget, and output gates, while the hyperbolic tangent $\tanh$ function was utilized to update the memory cell and generate the output value. These widely-used functions were selected to ensure reliable and consistent performance within the LSTM model.

3. **Training**: The LSTM model was trained on the training set using the backpropagation through time algorithm. The model parameters were optimized iteratively using the $Adam$ optimizer [38] with a learning rate of 0.001 to minimize the mean squared error loss function and enhance the prediction accuracy.

4. **Prediction and Evaluation**: The trained LSTM model was then used to make short-term traffic predictions on the testing set. The predictions were evaluated using RMSE, and MAPE.

4.4 ARIMA

The ARIMA model is, as mentioned before in Section 2, combination of AR, I, and, MA part. Each of these components is represented by a parameter in the ARIMA model, namely p, d, and q. The ARIMA's formula is as follows:

$$\begin{aligned}\text{ARIMA}(p,d,q):\\ x_t = c + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p}\\ + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q} + \varepsilon_t \quad (1)\end{aligned}$$

Here, the parameter p represents the order of the AR component, which captures the relationship between the current observation and a certain number of lagged observations. The coefficients ϕ_1, ϕ_2, ..., ϕ_p represent the weights assigned to each lagged observation. The parameter q represents the order of the MA component, which considers the dependency between the current observation and a linear combination of the residual errors from a moving average model applied to lagged observations. The coefficients θ_1, θ_2, ..., θ_q represent the weights assigned to the past residual errors. The constant term c captures the mean level of the time series. The term ε_t represents the error term, which represents the random component or noise in the model.

The parameter d in the model represents the degree of differencing applied to the time series, which transforms the data into a stationary series. Stationarity refers to the statistical properties of a time series remaining consistent over time, such as its mean and variance. By applying differencing, a non-stationary series can be transformed into a stationary series, making it easier to analyze and model. Differencing involves taking the difference between consecutive observations, demonstrated by the following formula:

$$\Delta x_t = x_t - x_{t-1}$$

where Δx_t represent differenced value at time t, and x_t and x_{t-1} represents the original observations at time t and $t-1$.

1. **Data Pre-processing**: Since tuning of parameter d involves the alteration of the original time series, this tuning process is considered data pre-processing. The original series, in Figure 5, does not seem to have a constant mean over time and variance does indeed seem to change significantly(e.g., the 3rd and 7th peaks show a difference of more than 40 vehicle reads). After taking the first order differencing, the series started to display a constant mean, with oscillations around 0, and to make sure the series is stationary, $Augmented\ Dick-Fuller$ (ADF) test is performed. The ADF test is a statistical test

used to determine whether a time series is stationary or not. The test results suggested that the differenced series is stationary, indicating that the first-order differencing was successful in removing the underlying trends and making the series suitable for modeling and forecasting purposes. Therefore, the time series required first-order differencing (d = 1) to achieve stationary data.

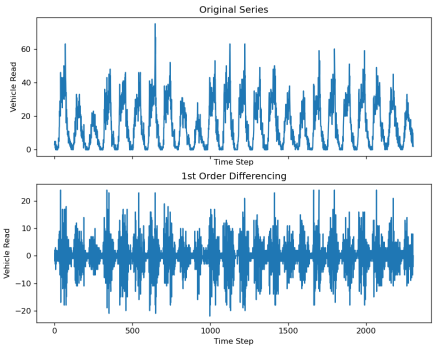

Figure 5: First Order Differencing of a Detector

2. **Model Configuration**: Determining the values of p, d, and q can be done using various methods. The approach selected in this research is to analyze the partial autocorrelation function (PACF) and autocorrelation function (ACF) plots. These plots provide insights into the correlation between a time series and its lagged values.

   To determine the appropriate value for the AR parameter (p), the PACF plot is examined. The PACF shows the correlation between the current observation and its specific lagged values while controlling for the correlation at shorter lags. Significant spikes in the PACF plot indicate the potential order of the AR component. As seen in Figure 6, there is only a significant spike in the first lagged value, hence 1 is an appropriate value for p.

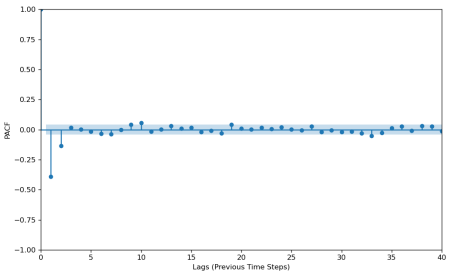

Figure 6: Partial Autocorrelation Function of a Detector

Similarly, the ACF plot is used to identify the order of the MA parameter (q). The ACF plot displays the correlation between the current observation and its lagged values without considering the correlation at shorter lags. Significant spikes in the ACF plot suggest the potential order of the MA component. As seen in Figure 7, there is only a spike in the first lagged value, hence 1 is an appropriate value for q.

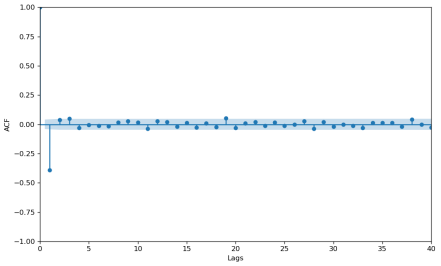

Figure 7: Autocorrelation Function of a Detector

By analyzing the plots of all the detectors, it is determined that the optimal values for p and q are 1 and 1, respectively.

3. **Training**: The ARIMA model was fitted to the training data, and the model coefficients were estimated using maximum likelihood estimation (MLE). The model was optimized to minimize the residual errors.
4. **Prediction and Evaluation**: The trained ARIMA model was used to make short-term traffic predictions on the testing set. The predictions were evaluated using the same evaluation metrics as in the LSTM model.

4.5 Prophet

The Prophet is sum of three functions of time plus an error term: growth function, seasonality function, holidays function , and error term [32]:

$$x_t = g(t) + s(t) + h(t) + e(t)$$

where:

- $g(t)$ represents the trend component (growth function), which captures non-periodic changes over time.
- $s(t)$ represents the seasonality component, which captures periodic changes that occur at regular intervals. These patterns can be daily, weekly, monthly, or yearly, and they often repeat in a predictable manner.
- $h(t)$ represents the effects of holidays or special events that may have an impact on the data but occur on irregular schedules. This component helps account for the influence of such events on the time series.
- $e(t)$ represents the residual or error component, which captures the unexplained and random fluctuations in the data. It includes any unexpected or unpredictable changes that are not accounted for by the other components of the model.

1. **Model Configuration**: For each of these components (except the error term), an appropriate functions need to be chosen.

   For the growth function, the logistic function was chosen due to its ability to model non-linear growth patterns commonly observed in time series data. It can be depicted by the following formula:

$$f(t) = \frac{L}{1 + e^{-k(t-t_0)}}$$

   where:

- $f(t)$ represents the predicted or forecasted value of the traffic volume at time t.
- L represents the upper bound or carrying capacity of the growth, which is the maximum value that the traffic volume can reach.
- k is the growth rate parameter that determines the steepness of the growth curve. A higher value of k implies a faster initial growth and a steeper curve.
- t_0 represents the inflection point or the time at which the growth curve transitions from increasing to decreasing. It indicates the time at which the peak traffic volume occurs.

The logistic growth function is fit for capturing the characteristic n-shaped curve seen in this study's data. This curve represents a gradual increase in traffic volume, followed by a peak, and then a gradual decrease [33]. Moreover, the logistic growth function has the advantage of boundedness, meaning it ensures that the forecasted values do not exceed certain limits. In the case of traffic data, it is logical to set a lower bound of zero and an upper bound based on the maximum observed value in the training data. This ensures that the forecasted traffic volumes remain within realistic and feasible ranges.

The parameter L is set to the maximum value seen by the detector in the training data, allowing the model to capture the upper bound of the growth. The remaining variables in the logistic growth function, including the growth rate k and the inflection point t_0, are automatically adjusted by the Prophet model based on the data patterns and trends.

The seasonality component in Prophet is modeled using Fourier series, which is a mathematical series that can approximate periodic functions by summing sine and cosine waves of different frequencies. The function for the seasonality component in Prophet can be expressed as follows:

$$f(t) = a_0 + \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi n t}{P}\right) + b_n \sin\left(\frac{2\pi n t}{P}\right) \right)$$

where:

- t represents the time variable.
- P is the period of the seasonality component.
- N is the number of Fourier terms used to model the seasonality.
- a_n and b_n are the coefficients associated with each Fourier term.

In this case, P is set to $1/96$ to capture the daily seasonality. Since there are 96 15-minute intervals in a day, setting P to $1/96$ scales the time to represent one day. This allows the model to capture the recurring patterns and variations that occur within a daily cycle. Moreover, the value of N is set to 3, which has been found to be effective in capturing both the daily and weekly data patterns [33]. By including three Fourier terms in the seasonality component, the model can capture the variation of the seasonal effects at different frequencies. a_n and b_n coefficients are automatically estimated by the Prophet model during the fitting process and are not explicitly set in the code snippet.

Furthermore, since there were no significant holidays observed in the Dutch calendar during the specific period of November 2019, no holiday components were included in the Prophet model for this research.

By configuring Prophet with the appropriate growth function, seasonality, and accounting for the absence of holidays, the model is equipped to effectively capture and forecast the complex patterns in the traffic data.

2. **Training**: The Prophet model was trained on the training set using an iterative optimization process. The model parameters were adjusted to minimize the defined loss function, thereby improving the accuracy of predictions [27].

3. **Prediction and Evaluation**: The trained Prophet model was utilized to make short-term traffic predictions on the testing set. The predictions were evaluated using the same evaluation metrics employed for the LSTM and ARIMA models.

By implementing and evaluating these three models using the described methodology, a comprehensive comparison analysis of LSTM, ARIMA, and Prophet for short-term traffic forecasting in Den Haag, Netherlands, can be performed. The following section presents the experiment and its results.

5 Experiment and Results

After training and configuring the LSTM, ARIMA, and Prophet models, they were evaluated on all the detectors in the test data. The average results across all detectors are presented below:

Model	RMSE	MAPE (%)
LSTM	7.13	33.19
ARIMA	5.5	25.77
Prophet	8.02	37.9

The table displays the average performance of each model in terms of RMSE and MAPE. The LSTM model achieved an average RMSE of 7.13 and an average MAPE of 33.19%. The ARIMA model yielded an average RMSE of 5.5 and an average MAPE of 25.77%. Finally, the Prophet model resulted in an average RMSE of 8.02 and an average MAPE of 37.9%.

To gain a better understanding of these results within the context of this study, the RMSE value of 7.13 indicates that, on average, the model's predictions deviated by 7.13 vehicles compared to the actual values.

These results provide an overview of the comparative performance of the three models in predicting short-term traffic. It is important to note that these values represent the average performance across all detectors in the test data. For a more detailed comparison of the models' performance over a specific detector, please refer to Figure 8.

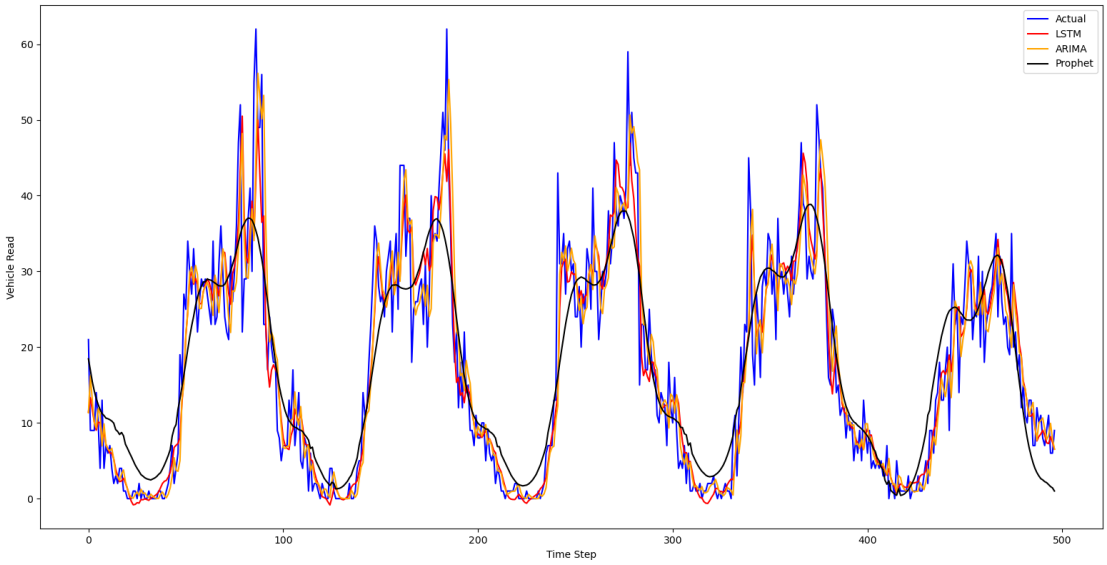

Figure 8: Performance comparison of LSTM, ARIMA, and Prophet models in forecasting a specific detector.

6 Comparison and Limitations

Based on the experimental results, ARIMA achieves the highest accuracy among the three models, while LSTM only outperformed Prophet.

To begin with Prophet, it has the poorest performance out of the three. As seen in Figure 8, it is not able to capture complexities and nuances as much as the other two models do. The main reason behind such a performance could be that the Prophet does not look for a causal relationship between the previous and current time steps, and instead, it tries to find the best curve to fit the data [39]. However, it must be noted that the Prophet is the only model, out of the three, that did not require data pre-processing. Given the motto of the Prophet being ease of use and of tuning [39], this performance is understandable

In Figure 8, it is observable that even though LSTM and ARIMA demonstrated decent performance in capturing certain patterns and fluctuations in the time series, there are some peaks (e.g. check the two of the highest peaks in Figure 8) that are predicted more accurately by ARIMA compared to LSTM. Such a result, demonstrating ARIMA outperforming LSTM, might very well contradict other research in this domain, such as [4] and [11], which clearly establish LSTM's dominance over ARIMA. This can be explained by two factors: data availability and the absence of spatial information in LSTM's architecture.

LSTM heavily relies on a large amount of data for effective training and capturing complex temporal relationships. On the other hand, while ARIMA benefits from more data to enhance its accuracy, it does not depend on data quantity as significantly as LSTM does, especially in the context of this particular study [34]. Other studies that resulted in LSTM's dominance used significantly larger datasets than this study. For example, this study includes 2880 data points for each detector, whereas [4] uses 25.11 million data points. The lesser availability of data might affect the accuracy of LSTM.

The second reason for LSTM's poorer performance when compared to ARIMA could be that there was no mechanism in LSTM's architecture that can utilize spatial information. Traffic flow is greatly influenced by location-specific factors such as road networks and congestion patterns. This relationship between the traffic flow and spatial information can be exploited by incorporating additional neural network layers such as convolutional neural network (CNN), as shown in [11], or correlation matrices such as origin destination correlation matrix (ODC), as shown in [4], which can help to capture spatial-temporal relationships in the data. However, LSTM which is used in this study primarily focuses on capturing temporal dependencies within the time series data. The lack of spatial information in LSTM's modeling approach may limit its ability to effectively capture and utilize location-based patterns, leading to comparatively lower accuracy in certain predictions.

It is also important to acknowledge that the performance of these models when compared with the MAPE values reported in other research studies [4, 6, 11, 28], has fallen below expectations.

The possible reason for lower performances in terms of MAPE values shown by these models can be mainly attributed to data availability. Researches that resulted in lower MAPE in their LSTM and ARIMA model utilized significantly higher amounts of data compared with this very study [4, 6, 11, 28]. This lesser availability of data affects the accuracy of LSTM, ARIMA, and Prophet.

As mentioned before, LSTM heavily relies on capturing long-term dependencies and patterns in data, the lower availability of data can significantly limit its ability to learn complex temporal relationships effectively [1,7]. LSTM models require a sufficient amount of training data to generalize well and make accurate predictions. With a smaller dataset, the

LSTM model may struggle to capture the underlying patterns and variations in the traffic flow data, leading to reduced performance. Overall it is a chronic situation of deep learning approaches where data abundance is needed for better training of the parameters [35].

With ARIMA, even though the impact of lesser data availability may not be as evident as with LSTM, it is important to note that a larger amount of data can still be beneficial for better training of the model's coefficients [16,34]. With a smaller dataset, the estimation of these coefficients may be less accurate, leading to less accurate predictions.

Prophet's ability to generate reliable forecasts relies on its ability to learn from historical patterns and make informed projections for the future [28]. A larger dataset provides more instances of these patterns, allowing the model to learn and generalize better [32]. In contrast, a smaller dataset may not provide enough representative examples, which can hinder Prophet's ability to capture the nuances of the underlying time series and make accurate predictions.

Furthermore, it is important to approach inflated MAPE values with caution. For instance, if the predicted value is 2 while the actual value is 1, it yields a 100% MAPE. However, in the context of traffic forecasting, such a small error of one vehicle may not have a substantial impact on the overall accuracy of ITS. Therefore, the relevance and significance of these higher MAPE values in the context of traffic forecasting need to be carefully considered.

7 Conclusion

In conclusion, this research paper aimed to address the challenges and knowledge gaps in short-term traffic forecasting by comparing three models: LSTM, ARIMA, and Prophet. The study focused on the context of Den Haag, Netherlands, and tried to answer the research question: *"Does LSTM outperform ARIMA and Prophet in the domain of short-term traffic forecasting?"*

Through experimental analysis, it was found that ARIMA achieved the highest accuracy among the three models, while LSTM outperformed Prophet. However, the performance of all models was lower than other researches' results in the domain when compared with MAPE values potentially due to limited data availability. Moreover, LSTM's sensitivity to data quantity and the absence of spatial information in its architecture, as well as Prophet's limitation in capturing complexities and establishing causal relationships, contributed to their inferior performance compared to ARIMA.

It is also important to note that the significance of higher MAPE values in the context of traffic forecasting should be approached with caution. Small errors in traffic prediction may not have a substantial impact on the overall accuracy of ITS.

The study highlights the need for further research to address the identified limitations and challenges. Future studies should explore approaches that leverage larger datasets and incorporate spatial information to enhance the accuracy and performance of traffic forecasting models. Additionally, the evaluation of more novel models against established methods should continue to ensure effectiveness and identify potential limitations in the domain of short-term traffic forecasting.

By advancing the understanding of different models' performance, this research contributes to the development of more accurate and reliable short-term traffic forecasting models, potentially supporting the improvement of Intelligent Transportation Systems.

8 Responsible research

The process of data collection and processing follows the responsible research principle of transparency. The data was collected from Den Haag Municipality, and as it is publicly available, it is accessible to everyone. Additionally, because the data consists only vehicle reads from the detectors, it is anonymous and therefore does not pose any ethical risks.

Moreover, the random initialization of weights in the LSTM model impedes this study to be exactly reproducible. To mitigate this potential impact of random fluctuations on the results, multiple iterations of the experiment are conducted and the mean result is taken. By employing this approach, it is hoped to improve the reliability of the findings and tried to ensure the reproducibility of the research. This adherence to responsible research practices strengthens the validity of this study.

References

[1] B. Lim and S. Zohren. "Time series forecasting with deep learning: a survey". arXiv:2004.13408 (2020).

[2] Y. Lv, Yisheng, Y. Duan, W. Kang, Z. Li and F.Y. Wang. "Traffic flow prediction with big data: A deep learning approach". In IEEE Transactions on Intelligent Transportation Systems (2014).

[3] A. Miglani and N. Kumar. "Deep learning model for traffic flow prediction in autonomous vehicles: A review, solutions, and challenges". In Vehicular Communications (2019).

[4] Zhao, Zheng, et al. "LSTM network: a deep learning approach for short-term traffic forecast". In IET Intelligent Transport Systems (2017).

[5] S. Bai, J. Zico Kolter, and K. Vladlen. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." arXiv:1803.01271 (2018)

[6] Williams, Billy M., and Lester A. Hoel. "Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results". In Journal of transportation engineering (2003).

[7] Tian, Yongxue, and Li Pan. "Predicting short-term traffic flow by long short-term memory recurrent neural network". In IEEE international conference on smart city (2015).

[8] 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, USA, 2018, pp. 57-65, doi: 10.1109/BigDataCongress.2018.00015. https://ieeexplore.ieee.org/document/8457731

[9] Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F. Y. (2015). Traffic Flow Prediction With Big Data: A Deep Learning Approach. IEEE Transactions on Intelligent Transportation Systems, 16(2), 865-873.

[10] Van Hinsbergen, C. P. I. J., Van Lint, J. W. C., Sanders, F. M. (2007). Short Term Traffic Prediction Models. Transportation Research Part A: General, 25(2-3), 139-151.

[11] Yu, B., Yin, H., Zhu, Z. (2018). Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence (pp. 3634-3641).

[12] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results" Journal of transportation engineering, vol. 129, no. 6, pp. 664-672, 2003.

[13] Nikovski, D., et al. Univariate short-term prediction of road travel times. in Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems. 2005

[14] Huisken, G. and E.C.v. Berkum. A comparative analysis of short-range travel time prediction methods. in Transportation Research Board 82nd Annual Meeting. CD-ROM. 2003.

[15] Levin, M., Tsao, Y.D.: "On forecasting freeway occupancies and volumes(abridgment)", Transport. Res. Record, 1980, 773, pp. 47-49

[16] Ahmed, M.S., Cook, A.R.: "Analysis of freeway traffic time-series data byusing Box-Jenkins techniques". Transport. Res. Record, No. 722, 1979

[17] Ben-Akiva, M., et al. "Real-time prediction of traffic congestion. in Vehicle navigation information systems: conference record of papers." 1992

[18] Zhang, Y., Liu, Y.: 'Traffic forecasting using least squares support vectormachines', Transportmetrica, 2009, 5, (3), pp. 193–213

[19] Lv, Y., Duan, L., Kang, W., et al. (2014). Short-term traffic flow forecasting using deep belief networks. Transportation Research Part C: Emerging Technologies, 44, 103–120.

[20] Gao, L., Hu, S., Lu, J. (2013). Short-term traffic flow prediction using k-nearest neighbor method. Journal of Advanced Transportation, 47(4), 368-384.

[21] Wang, F., Li, Z., Zhou, Y., Zhang, H. (2018). Short-term traffic flow forecasting with support vector regression based on wavelet decomposition and grid search optimization. Journal of Advanced Transportation, 2018, 1-17. doi: 10.1155/2018/1424768

[22] Jiang, Y., Ye, J., Zhao, Y., Zhang, Y. (2019). Data imputation for short-term traffic forecasting: A deep learning approach. Transportation Research Part C: Emerging Technologies, 99, 18-33.

[23] Wang, J., Zhang, C. (2021). A novel traffic prediction method based on long-short term memory network and external factors. IEEE Transactions on Intelligent Transportation Systems.

[24] Zheng, Y., Zhou, X., Li, X., Wang, D. (2020). A deep learning approach to traffic flow prediction using spatial-temporal correlations. Transportation Research Part C: Emerging Technologies, 113, 482-497.

[25] Zhang, J., Ma, W., Du, H., Luo, X. (2020). A hybrid traffic forecasting model based on LSTM and kriging with external variables. IEEE Access, 8, 64935-64946.

[26] Yu, L., Ma, T., Liu, X. (2019). Real-time traffic flow prediction with spatial-temporal correlations and multivariate time series. IEEE Transactions on Intelligent Transportation Systems, 20(3), 888-898.

[27] Taylor, S. J., Letham, B. (2017). Forecasting at scale. The American Statistician.

[28] N. K. ChikkaKrishna, P. Rachakonda and T. Tallam, "Short - Term Traffic Prediction Using Fb-PROPHET and Neural-PROPHET," 2022 IEEE Delhi Section Conference (DELCON), New Delhi, India, 2022, pp. 1-4, doi: 10.1109/DELCON54057.2022.9753459.

[29] Brownlee, J. (2020). How to Develop LSTM Models for Time Series Forecasting. Retrieved from https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/.

[30] Brownlee, J. (2022). Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras. Retrieved from https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/.

[31] Keras. LSTM Layer. Retrieved from https://keras.io/api/layers/recurrent_layers/lstm/.

[32] Krieger, M. (2021). Time Series Analysis with Facebook Prophet: How It Works and How to Use It. Towards Data Science. Available online: Time Series Analysis with Facebook Prophet.

[33] Facebook. Prophet: Quick Start. Retrieved from https://facebook.github.io/prophet/docs/quick_start.html

[34] Alghamdi, Taghreed Elgazzar, Khalid Bayoumi, Magdi Sharaf, Taysseer Shah, Sumit. (2019). Forecasting Traffic Congestion Using ARIMA Modeling. 1227-1232. 10.1109/IWCMC.2019.8766698.

[35] Marcus, G. "Deep Learning: A Critical Appraisal." arXiv:1801.00631 (2018).

[36] Tian, Yan, et al. "LSTM-based traffic flow prediction with missing data." In Neurocomputing (2018).

[37] Anirudh Ameya Kashyap, Shravan Raviraj, Ananya Devarakonda, Shamanth R Nayak K, Santhosh K V Soumya J Bhat — Fabio Galatioto (Reviewing editor) (2022) Traffic flow prediction models – A review of deep learning techniques, Cogent Engineering, 9:1, DOI: 10.1080/23311916.2021.2010510

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980, (2014)

[39] Goled, S. (2021). Why are people bashing Facebook Prophet. Analytics India Magazine. https://analyticsindiamag.com/why-are-people-bashing-facebook-prophet/