

Arrival Time Predictions for Buses using Recurrent Neural Networks

Ankomsttidsprediktioner för bussar med rekurrenta neurala nätverk

Christoffer Fors Johansson

Supervisor : Mattias Tiger
Examiner : Fredrik Heintz

External supervisor : Simon Johansson

Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>.

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: <http://www.ep.liu.se/>.

Abstract

In this thesis, two different types of bus passengers are identified. These two types, namely current passengers and passengers-to-be have different needs in terms of arrival time predictions. A set of machine learning models based on recurrent neural networks and long short-term memory units were developed to meet these needs. Furthermore, bus data from the public transport in Östergötland county, Sweden, were collected and used for training new machine learning models. These new models are compared with the current prediction system that is used today to provide passengers with arrival time information.

The models proposed in this thesis uses a sequence of time steps as input and the observed arrival time as output. Each input time step contains information about the current state such as the time of arrival, the departure time from the very first stop and the current position in Cartesian coordinates. The targeted value for each input is the arrival time at the next time step. To predict the rest of the trip, the prediction for the next step is simply used as input in the next time step.

The result shows that the proposed models can improve the mean absolute error per stop between 7.2% to 40.9% compared to the system used today on all eight routes tested. Furthermore, the choice of loss function introduces models that can meet the identified passengers need by trading average prediction accuracy for a certainty that predictions do not overestimate or underestimate the target time in approximately 95% of the cases.

Acknowledgments

I would like to take the opportunity in this section and acknowledge some individuals who have been helpful, inspiring and encouraging through this thesis and in the past few years leading up to this point.

First of all, I want to direct my gratitude and acknowledgement to my supervisor **Mattias Tiger** and examiner **Fredrik Heintz**. Thank you for the opportunity to conduct this thesis and for taking your time and sharing your comprehensive knowledge. Both of you are very inspirational and exemplary professionals.

I would also like to thank all employees at *Attentec AB* for the warm welcome. I have completely lost count on the number of interesting and fruitful discussions and ideas as well as the number of much-needed coffee breaks I have spent with you. A special thanks to my external supervisor **Simon Johansson**. You welcomed me in the best way possible and always had time to discuss matters. Also, thank you to **Albin Furin** for taking your valuable time and helping me to get started with AWS.

Finally, I would like to thank my classmates **Hampus Carlsson** and **Anton Hölscher**. I think we formed a unique environment during our time at the university where we somehow combined a lot of interesting discussions with hard work and great fun. Thank you for that.

Contents

Abstract	iii
Acknowledgments	iv
Contents	v
List of Figures	vii
List of Tables	ix
1 Introduction	1
1.1 Motivation	1
1.2 Aim	2
1.3 Research questions	2
1.4 Delimitations	2
2 Background	3
2.1 Arrival time predictions	3
2.2 Linear regression	4
2.3 Neural Networks	7
2.4 Recurrent Neural Networks	10
2.5 Regularization	14
2.6 Gaussian processes	14
2.7 Evaluation techniques	15
3 Data	17
3.1 GTFS	18
3.2 GTFS Realtime	19
3.3 Stops	19
3.4 Trips	20
3.5 Routes	25
3.6 Baselines	26
4 Method	28
4.1 Data selection	28
4.2 Loss functions	29
4.3 Models	30
4.4 Training	32
4.5 Feature vector	33
4.6 Predictions	34
4.7 Frameworks and hardware	35
5 Results	36

5.1	Arrival time predictions	36
5.2	Summary	54
6	Discussion	55
6.1	Analysis of the results	55
6.2	Loss functions	56
6.3	Method	57
6.4	Applicability	57
7	Conclusion	59
7.1	Future work	60
	Bibliography	62

List of Figures

2.1	Linear regression with made up data.	6
2.2	Linear regression with the input expressed as all polynomial combinations to different degrees.	7
2.3	Example of a feedforward NN with one hidden layer. $n = 2$, $M = 3$ and $K = 1$	9
2.4	An unfolded RNN.	11
2.5	Block diagram of a LSTM unit.	12
3.1	Overview of the data flow. Each bus is equipped with a GPS transmitter estimating the current position. This information along with calculated arrival, departure and prognosis-information from ÖT is automatically propagated to Trafiklab, which in turn make the information available through a set of APIs.	17
3.2	Simplified schema of the relations between individual files in the GTFS feed available at Trafiklab.	18
3.3	Illustration over the bus stop locations. Each red circle corresponds to a geographic location labelled as a type of stop. A stop in this figure is included in or several means of transport, including bus, train, tram, or boat.	19
3.4	Visualization of a trip. The blue line represents the planned trajectory. The red and green circles are stops on the trip where red circles represents timed stops.	20
3.5	Observed arrival and departure times compared to the schedule for a trip from Linköping to Norrköping, 16.00 local time 2019-02-18. Bold bus stops are timed stops.	21
3.6	Observed arrival and departure times compared to the schedule for a trip from Linköping to Norrköping, 16.00 local time 2019-02-26.	22
3.7	MSE and MSE^* for the trip from Linköping to Norrköping 16.00 local time at two different dates.	23
3.8	Observed arrival and departure times compared to the schedule for a trip from Linköping to Norrköping, 16.00 local time during the period 2019-02-11 to 2019-03-04, $N = 13$	24
3.9	MSE for a trip from Linköping to Norrköping, 16.00 local time during the period 2019-02-11 to 2019-03-04, $N = 13$	25
3.10	Illustration of how the CPS operates.	26
4.1	Trajectory for the routes for which LSTM models were developed.	28
4.2	A plot of the loss functions examined.	30
4.3	Network architecture of the examined models.	31
5.1	MSE for trips from Linköping to Norrköping when the schedule is predicted by observations.	37
5.2	MSE for trips from Norrköping to Linköping when the schedule is predicted by observations.	37
5.3	Prediction errors on the test set with 159 sequences as measured in MAE . The LSTM model is trained with MSE as loss.	38

5.4	Type of error at each stop for the CPS and TBP on route 70.	39
5.5	Type of error at each stop on route 70.	39
5.6	MSE for trips on route 303 when the schedule is predicted by target values.	40
5.7	MSE for trips on route 303 when the schedule is predicted by target values.	40
5.8	Prediction errors on the test set with 409 sequences as measured in <i>MAPE</i>	41
5.9	Type of error at each stop for the CPS and TBP on route 303.	41
5.10	Type of error at each stop on route 303.	42
5.11	MSE on route 616 for trips from Linköping to Borensberg when the schedule is predicted by target values.	42
5.12	MSE on route 616 for trips from Borensberg to Linköping when the schedule is predicted by target values.	43
5.13	Prediction errors on the test set with 108 trips as measured in <i>MAPE</i>	43
5.14	Type of error at each stop for the CPS and the LSTM model loss function with Pang et al. on route 616.	44
5.15	MSE on route 20, located in Linköping when the schedule is predicted by target values.	44
5.16	MSE on route 20, located in Linköping when the schedule is predicted by target values.	45
5.17	Prediction errors on the test set with 145 trips as measured in <i>RMSE</i>	45
5.18	Type of error at each stop for the CPS and the LSTM model with loss function Pang et al. on route 20.	46
5.19	MSE on route 119, located in Norrköping when the schedule is predicted by target values.	46
5.20	MSE on route 119 for trips in Linköping when the schedule is predicted by target values.	47
5.21	Prediction errors on the test set with 716 trips as measured in <i>RMSE</i>	47
5.22	Type of error at each stop for the CPS and the LSTM model with loss function <i>MSE</i> on route 119.	48
5.23	MSE on route 3, located in Linköping, when the schedule is predicted by target values.	48
5.24	MSE on route 3, located in Linköping, when the schedule is predicted by target values.	49
5.25	Prediction errors on the test set with 1478 trips as measured in <i>RMSE</i>	49
5.26	Type of error at each stop for the CPS and the LSTM model with <i>MSE</i> as loss function on route 3.	50
5.27	MSE on route 450 from Norrköping to Söderköping, when the schedule is predicted by target values.	50
5.28	MSE on route 450 from Söderköping to Norrköping, when the schedule is predicted by target values.	51
5.29	Prediction errors on the test set with 104 trips as measured in <i>MAPE</i>	51
5.30	Type of error at each stop for the CPS and the LSTM model with loss function <i>MSE</i> on route 450.	52
5.31	MSE on route 45 from Norrköping to Söderköping, when the schedule is predicted by target values.	52
5.32	MSE on route 45 from Söderköping to Norrköping, when the schedule is predicted by target values.	53
5.33	Prediction errors on the test set with 231 trips as measured in <i>RMSE</i>	53
5.34	Type of error at each stop for the CPS and the LSTM model with loss function Pang et al. on route 45.	54

List of Tables

3.1	MSE and MSE* for the example in Figure 3.5.	22
3.2	MSE and MSE* for the example trip in Figure 3.6.	22
3.3	Error function results for the example trip with $N = 13$ and $S = 14$	25
3.4	Comparison of characteristics between two datasets.	25
4.1	Hard measures on the selected routes from the large dataset.	29
4.2	Loss functions evaluated in this thesis.	31
4.3	Static training parameters and their values.	32
4.4	Training parameters grouped by route.	33
5.1	Evaluation results for route 70, $N = 159$	38
5.2	Evaluation results for route 303, $N = 409$	41
5.3	Evaluation results for route 616, $N = 108$	43
5.4	Evaluation results for route 20, $N = 145$	45
5.5	Evaluation results for route 119, $N = 716$	47
5.6	Evaluation results for route 3, $N = 1478$	49
5.7	Evaluation results for route 450, $N = 104$	51
5.8	Evaluation results for route 45, $N = 231$	53
5.9	Summary of the results.	54

List of abbreviations

API	<i>Application Programming Interface</i>
AVL	<i>Automatic Vehicle Location</i>
BPTT	<i>Backpropagation Through Time</i>
CPS	<i>Current Prediction System</i>
CPU	<i>Central Processing Unit</i>
GPU	<i>Graphical Processing Unit</i>
GTFS	<i>General Transit Feed Specification</i>
GTFS-RT	<i>General Transit Feed Specification Realtime</i>
LSTM	<i>Long Short-Term Memory</i>
MAE	<i>Mean Absolute Error</i>
MAPE	<i>Mean Absolute Percentage Error</i>
MSAP	<i>Multi-Step Ahead Prediction</i>
MSE	<i>Mean Squared Error</i>
ML	<i>Machine Learning</i>
NN	<i>Neural Network</i>
RAM	<i>Random Access Memory</i>
RMSE	<i>Root Mean Squared Error</i>
RNN	<i>Recurrent Neural Network</i>
SGD	<i>Stochastic Gradient Decent</i>
TBP	<i>Timetable Based Predictions</i>
ÖT	<i>Östgötatrafiken</i>



1 Introduction

Public transport is and will be an important part of modern cities. More and more people move to densely populated areas and reliable, accurate public transport is something that benefit us all.

In 2018, approximately 1.6 billion boardings were made in the regional line traffic in Sweden [1]. These boardings include transport by bus, train, tram and ship. The bus was the most used means of transport and accounted for about 52% of all boardings. People use public transport to get to work, school and leisure activities, making it an important part of society from economic, environmental and social perspectives.

Östgötatrafiken AB (ÖT) is responsible for the public transport in Östergötland county, Sweden. They want to provide an easy, comfortable and reliable alternative to taking the car. One step in that direction is to provide accurate information about the estimated time of arrival for their means of transportation.

Recent work shows that Machine Learning (ML) approaches can predict arrival times for buses with promising results [2] [3] [4]. ÖTs buses are equipped with a 1 Hz GPS transmitter that can be tracked in real-time trough an interactive map [5]. This data can be used as a basis for learning to predict the arrival time at various bus stops. Pang et. al [3] showed that a recurrent neural network (RNN) with a long short-term memory (LSTM) block together with static information about the world and GPS data, can predict arrival times with higher accuracy than previous methods, even for several bus stop s ahead of the current one. They used GPS data with ~ 0.033 Hz from buses in Beijing.

In this thesis we propose an ML approach to arrival time prediction for ÖT which performs better on multiple accounts compared to the current prediction system in use by ÖT.

1.1 Motivation

There are some aspects that are interesting to take into consideration regarding public transportation today and in the future. For instance, Sassen [6] describes that urbanization looks different in different parts of the world, but the general trend is the same. People move to denser areas which will put demands on smart transport solutions in our cities to cope with the increasing concentration of people. There are also environmental reasons for making the public way of travelling more attractive. Public transport is a good way to reduce emissions of fossil fuels compared to individual car journeys. A recent study [7] shows that the service

quality has a direct effect on peoples intention to utilize public transport. Waiting time is identified as one of the most valued variable in determining the quality of service [8], and good arrival time predictions will facilitate the planning of the trip for travellers.

What a good arrival time prediction is depends on the situation. For instance, passengers already on board a bus might be interested in a prediction closer to the worst case. Such predictions can give a time of when the passengers reach their destination at the latest, so that the plans after the arrival can be adjusted accordingly. People who are planning to go by a bus on the other hand, might be interested in a prediction that does not overestimate the time until departure to prevent being abandoned at the bus stop. Also, being too conservative with the arrival time predictions at bus stops risk causing people to wait unnecessarily long. Today, ÖT base their arrival time predictions on statistical models from historical data. More data, available in real-time, and progress in the ML research area opens up for improvements and new approaches to the problem of arrival time predictions for buses.

This thesis has been conducted with Attentec in collaboration with ÖT. Attentec is a consultant firm specialized on new technology in the domains of internet of things and streaming media. Attentec's interest in problem solving with modern techniques align well with ÖTs need to provide better arrival time predictions.

1.2 Aim

The goal of this thesis is to study how better arrival time predictions can be made compared to existing systems by using modern techniques and recent progress in the field of machine learning. It includes and requires investigating what better arrival time predictions actually means and for whom.

1.3 Research questions

Considering the aim of the thesis and the current research in ML in general and arrival time predictions for buses using ML techniques in particular, the following set of research questions are studied in this thesis.

1. How well can a recurrent neural network model predict arrival times compared to the existing prediction system used by Östgötatrafiken?
2. What cost functions and validations are suitable for arrival time predictions?
3. The current passenger and the passenger-to-be have different needs in terms of predictions. How can this be facilitated?

1.4 Delimitations

The answers to the research questions might be delimited to external factors that ultimately influences the results. This is the section where such delimitations should be discussed. One such delimiting factor is the time span of the dataset, which only includes approximately two months of the whole year. Analysis made on this dataset will therefore not capture how the prediction system performs with respect to seasonal events, such as vacations during the summer or the first heavy snowfall of the year.

Eight routes were selected as a result of the discussions with ÖT to span a variety of highly interesting settings. The results in this thesis is based solely on these eight, which may limit the scope of the conclusions.



2 Background

Arrival time predictions using learning techniques is an ongoing research area, and there are different ways to approach the problem. The purpose of this chapter is to provide context and definitions to important concepts related to the research questions.

2.1 Arrival time predictions

Predicting when a bus will be at a certain location is a severe task because of the many stochastic variables involved and their complex relations. Weather, traffic intensity, road work, accidents, intersections and the number of passengers are some examples of factors that can affect how long it takes to reach a destination. It can be reasonably assumed that the arrival time is the output from a function with the affecting variables as input. Yu et al. [9] describes how a support vector machine (SVM) regression model could be used to predict arrival times for buses in Hong Kong back in 2011. The authors used travel times together with weather data as a basis for their model.

SVMs are not able to fit a non-linear function unless the kernel trick [10] is introduced as shown by Boser, Guyon and Vapnik [11]. One problem with their model is the scaling ability to large problems, since the kernel matrix grows quadratically with the number of training samples [12].

Models based on historical data are not very adaptive, and unexpected events in traffic can not be taken into consideration by models based on historical events. One idea to overcome this problem is to weigh in the current state of the world in the prediction. Zhou, Zheng, and Li [13] used data from mobile phone application users connected to their prediction system in order to gain information about the worlds current state, and combined it with historical information. To overcome the problem of the prediction system being dependent on active, travelling users, Gong, Liu, and Zhang [14] utilized an automatic vehicle location (AVL) system with global positioning system (GPS) data as a basis for their three proposed models.

AVL data makes it possible to build models based on measured historical conditions. An alternative approach is to use simulations as a basis for prediction models. For instance, Ben-Akiva et al. [15] proposed such a prediction system back in 1998. However, models based on historical data have become the more popular alternative, where particularly deep learning models have shown promising results [16].

In this thesis, a regression approach toward the problem is taken. Particularly, a variant of *Neural Networks* (NNs) is used to capture long-range dependencies over time, namely *Recurrent Neural Networks* (RNN). The RNN model will be approached in this chapter by starting to examine the concept of linear regression.

There are mainly two reasons for why NNs are of interest to this thesis. The first one is the previous success in multi-step ahead prediction (MSAP) models for arrival time predictions in the past. Chien, Ding, and Wei [17] proposed two models back in 2002 based on NNs where one model accumulates the travel times throughout the trip while the other method only considered data between two consecutive stops. Another study on a transit bus route in Houston by Jeong and Rilett [18] suggests that the NN model outperformed the multi-variable linear regression approach as well as using a simple historical average approach. In a more recent study, Chen [19] proposes an approach where several NNs is randomly trained with promising results, indicating that the NNs are still indeed useful in this domain. The second reason for why NN are of interest to this thesis is that it is a good basis for understanding RNNs as well as the variant of RNNs that eventually became the basis for the models in this thesis.

2.2 Linear regression

One way to approach NNs and understand them, as suggested by Goodfellow, Bengio and Courville [20], is to start off by considering linear models such as linear regression. This section intends to follow that approach by providing sufficient background on linear regression so that the limitations as well as ways to overcome them becomes evident.

Linear regression aims to solve a regression problem, such that a system is able to produce a scalar prediction $\hat{y} \in \mathbb{R}$ on the output $y \in \mathbb{R}$, given the vector of input features $\mathbf{x} \in \mathbb{R}^n$. Each individual feature is associated with a weight w so that the influence or importance of that particular feature could be adjusted. For instance, the weight \mathbf{w}_i could be a positive number, suggesting that a large value of the feature x_i would increase the final prediction. Similarly, a negative value reduces the prediction for larger values of the feature. A value close to zero for \mathbf{w}_i would reduce the importance of the feature x_i [20]. This is how the importance and influence of features are adjusted. The problem now is to decide the values of the weights that gives the best result, which will be examined later on. For now, the output could be defined as

$$y = \mathbf{w}^T \mathbf{x} + b. \quad (2.1)$$

As anticipated based on the task of predicting y , the prediction is expressed by $\hat{y} = \mathbf{w}^T \mathbf{x} + b$. The bias term b introduces the possibility for the model to skip passing the line through the origin [20]. Note that bias in this case refers to the model being biased towards b when no input is present as opposed to the statistical interpretation of the bias term.

Now, Bishop [21] suggests that the simplest linear regression model is the one modelling the input variables as a linear combination of the inputs expressed by

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \dots + w_n x_n \quad (2.2)$$

where \mathbf{x} and \mathbf{w} is the n -dimensional vectors previously described. The parameter w_0 is the equivalent of b in Equation 2.2. Bishop [21] further describes the linear combination of the adjustable weights \mathbf{w} as the most important property of the linear regression model. This implies that the model is a linear combination of the input variables \mathbf{x} as well. This linearity could however be expressed through a non-linear *basis function* $\Phi(\mathbf{x})$, where $\Phi = (\phi_0, \dots, \phi_{N-1})^T$ and N is the number of parameters in the model. If a non-linear basis function is used, the function $y(\mathbf{x}, \mathbf{w})$ becomes non-linear to the input while the linearity in \mathbf{w}

remains. Equation 2.3 illustrates how $y(\mathbf{x}, \mathbf{w})$ is expressed using a basis function. Note that w_0 is not present in the equation and the bias is instead handled by defining $\phi_0 = 1$.

$$y(\mathbf{x}, \mathbf{w}) = \sum_{i=0}^{N-1} w_i \phi_i(x_i) = \mathbf{w}^T \Phi(\mathbf{x}) \quad (2.3)$$

There are some alternatives when it comes to choosing a suitable basis function. As an example, Bishop [21] describes a basis function termed the Gaussian basis function defined as

$$\phi_i(x) = \exp\left(-\frac{(x - \mu_i)^2}{2\sigma^2}\right) \quad (2.4)$$

where μ is a vector with points in the feature space with σ adjusting their scale.

Recall that the weight vector is a crucial part of the model. In theory, the weights should steer the input to the target output. This is where the learning part of the algorithm is introduced. First of all, some kind of penalty, loss or cost function should be defined so that each prediction could be evaluated towards the target output, and there are a few alternatives. Goodfellow, Bengio and Courville [20] suggests *mean squared error* (MSE) defined by Equation 2.5 for n predictions.

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n} \quad (2.5)$$

The cost of the model should be minimized. Since MSE now represents the cost and evaluates the model, the goal now is to maximize the performance of the model by minimizing the cost function. Recall that \hat{y} can be expressed similar to Equation 2.1, which means that the MSE can be rewritten as in Equation 2.6.

$$MSE = \frac{\sum_{i=1}^n (y_i - (\mathbf{w}^T \phi(x_i)))^2}{n} \quad (2.6)$$

The reason for rewriting MSE is to end up with an equation for which MSE could be minimized with respect to \mathbf{w} and b . One way to do so is to solve the gradient for zero with respect to \mathbf{w} and b [20] if all the points are present at the time of evaluation. Alternatively, n points at a time is considered, creating a continuous stream called *sequential learning*. The weights are then updated step by step with a method called *stochastic gradient descent* (SGD) [21]

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla MSE_n \quad (2.7)$$

where τ denoted the iteration and η the learning rate. Note that this works for other cost functions satisfying the requirements even though MSE is specified in Equation 2.7. SGD is explained and used further in this chapter so let us consider how the MSE could be minimized by solving the partial derivatives equal to zero with respect to \mathbf{w} and b as in Equation 2.8.

$$\frac{\partial MSE}{\partial w} = \frac{\partial MSE}{\partial b} = 0 \quad (2.8)$$

Solving Equation 2.8 for \mathbf{w} and b results in Equation 2.9 and Equation 2.10, respectively. The bar in the equations denotes the mean value. Thus, $\bar{x} = \frac{x_1 + \dots + x_n}{n}$ and $\bar{y} = \frac{y_1 + \dots + y_n}{n}$. Similarly, the notation $\overline{xy} = \frac{xy_1 + \dots + xy_n}{n}$ and $\overline{x^2} = \frac{x_1^2 + \dots + x_n^2}{n}$ for n points.

$$w = \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - (\bar{x})^2} \quad (2.9)$$

$$b = \bar{y} - w\bar{x} \quad (2.10)$$

The weight and bias could then directly be calculated for a set of input and targeted output data.

To make it concrete, consider the simple linear regression example with one assumed independent variable x in Figure 2.1. Figure 2.1a depicts the data used in this example. The data is randomly divided into two sets, namely train and test. The train set consists of 75% of the data points while the remaining 25% of the points is considered to be test points. The purpose of creating two different sets is to be able to test how well the model performs on previously unseen data or in other words, how good the model *generalizes* [21].

Figure 2.1b illustrates a linear regression model using the following basis function

$$\phi_i(x) = x^i \quad (2.11)$$

with $i = 0$ and $i = 1$, creating a constant and a first order polynomial, respectively. Intuitively the first degree polynomial is better to capture the pattern in the data which also is confirmed by the R^2 score. R^2 is a useful statistical measure for measuring how close the fitted regression line is to the data [22] defined as

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2} \quad (2.12)$$

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i. \quad (2.13)$$

Not surprisingly, the first degree polynomial fits the data much better than the constant according to the R^2 score.

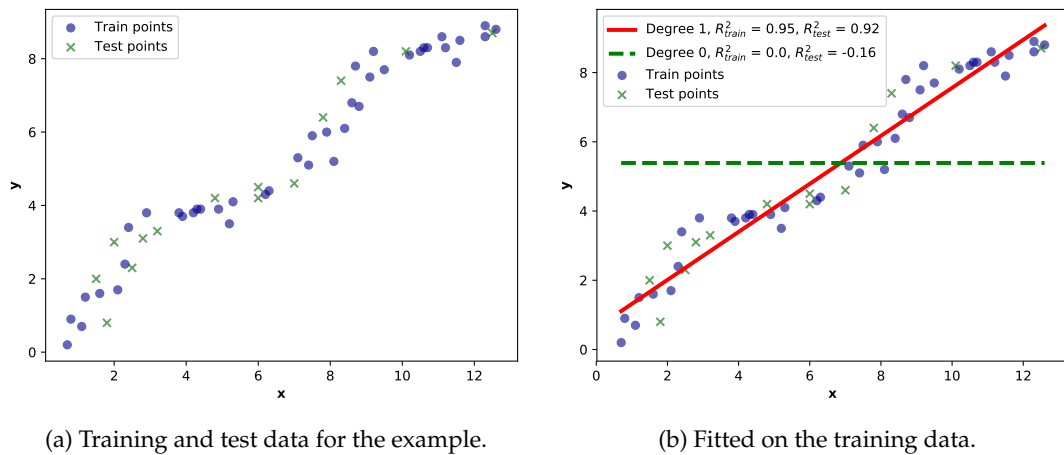


Figure 2.1: Linear regression with made up data.

The tiny fluctuation present in the data could perhaps be captured even better with greater polynomials utilizing the same basis function as in Equation 2.11. It appears that a polynomial of degree four perhaps could capture the two oscillations quite well. Recall that the model still is based on linear regression while the input is expressed as all polynomial combinations up to a specified degree. Let us as an experiment and to test the hypothesis, express

the input data as all polynomial combinations up to a few interesting polynomials. The result is illustrated in Figure 2.2 where the train data is visible to the left in Figure 2.2a while the test data is present on the right hand side in Figure 2.2b. As expected, the polynomial of fourth degree is able to fit both the train and test data best of the variants considered, closely followed by the third degree polynomial.

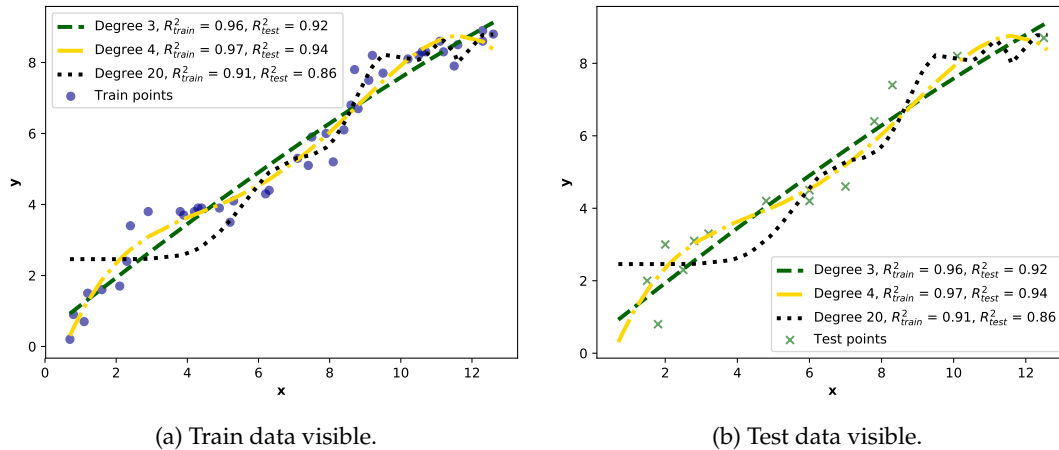


Figure 2.2: Linear regression with the input expressed as all polynomial combinations to different degrees.

Before moving the focus towards NNs one more important phenomena present in the example is worth to mention. One of the goals with machine learning is to generalize and perform well on previously unseen data [20]. For that purpose, the dataset was divided into a training set and a test set. Now, if the model performs poorly on the training data, just as the constant line in Figure 2.1b, the model is said to be *underfitted* and is not able to capture the patterns in the data. Furthermore, if the performance on the train data is fine while the test data is poor, the model is *overfitted*. It means that the model is too heavily biased toward the training data that it will perform bad on new data [20]. Both underfitting and overfitting is important to be aware of so that the model is able to learn something without just becoming an expert on predicting already seen data. Commonly, the training set consist of about 80% of all data while the test set holds the remaining 20% [20]. Thus, the model can observe an gain experience from around 80% of the available data.

2.3 Neural Networks

By starting to consider linear regression, a handful of useful concepts in machine learning were taken up along the way. This section about NNs builds on that previous knowledge.

The term NNs is broad and covers a wide set of models [21]. However, the most typical set of NN models is called *deep feedforward networks* or *multilevel perceptions* (MLPs). They are called feedforward since the information flow goes from a certain input x , through intermediate computations to the target output y without feeding output back to itself [20]. Networks with feedback on the other hand is included in the concept of RNNs and is further described in section. 2.4.

In the previous section where linear regression was considered, it was stated that the model were limited to functions as linear combinations of the input. The non-linear basis function makes the regression model non-linear. However, the model is still linear in the model parameters. Another problem is that a linear regression model can not recognize interaction between two input variables [20]. In order to represent linear models as non-linear functions of x , Goodfellow, Bengio and Courville [20] describes two approaches. The first

one is similar to what was presented previously using a transformed input $\Phi(\mathbf{x})$. The second equivalent way is to apply the kernel method [21] to gain the non-linearity for our learning algorithm where the Φ mapping is implicit. Now, the challenge is to figure out how the mapping Φ should be done. One way is to use a very generic Φ of high dimension so that there are enough parameters to fit a function against all points in the training data. However, this approach turned out to cause problems with generalization as shown in section 2.2. Another idea is to let humans engineer a good mapping which was the dominant approach before *deep learning*. It turns out that finding a suitable mapping Φ is domain specific and requires a lot of work in each domain. Instead, the idea of deep learning is to learn using a very flexible class of Φ [20].

Recall from section 2.2 that Equation 2.3 describes how the linear regression model could utilize a non-linear basis function $\Phi(\mathbf{x})$. Now, the aim is to extend that equation so that the basis function contains adjustable parameters together with the already adjustable weights. The values of these parameters and the weights will be decided during the training stage later on [21]. Equation 2.14 describes the first step in a NN which is the M linear combinations of the inputs x_1, \dots, x_N for $j = 1, \dots, M$. The weights in the first layer is denoted as $w_{ji}^{(1)}$ and the biases as $w_{j0}^{(1)}$. The result from Equation 2.14 is passed through a function and then propagated forward and used as input. This can be thought of as *layers*, since the methodology repeats itself all the way to the last layer where the output is produced. The number of layers is dependent on the type of problem. For instance, only two layers are necessary if the input is linearly separable. However, the NN could easily be extended with another layer which introduces the possibility to capture non-linear patterns.

$$a_j = \sum_{i=1}^n w_{ji}^{(1)} x_i + w_{j0}^{(1)} \quad (2.14)$$

The result from Equation 2.14 is called *activation* and is passed through a non-linear *activation function* φ shown in Equation 2.15 to create the output from the *hidden units*.

$$h_j = \varphi(a_j) \quad (2.15)$$

Similarly, the output from the hidden units is passed to the next, second layer of the network to once again create linear combinations as described by

$$a_k = \sum_{j=1}^M w_{kj}^{(2)} z_j + w_{k0}^{(2)}. \quad (2.16)$$

This time, the variable k refers to the total number of outputs, such that $k = 1, \dots, K$. The weights and biases in this second layer is denoted by $w_{kj}^{(2)}$ and $w_{k0}^{(2)}$, respectively. The activation function φ in the regression case is usually the following identity

$$y_k = a_k \quad (2.17)$$

so that values can be unbounded throughout the network [21].

Figure 2.3 depicts an NN structure, with three dimensional input, one hidden layer and one output layer with one output to give an intuition of how a NN architecture could look like.

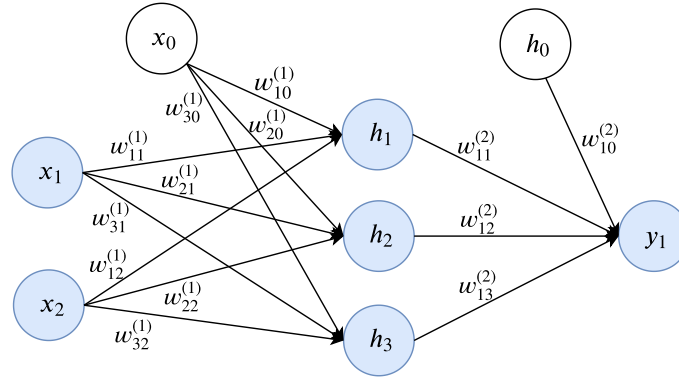


Figure 2.3: Example of a feedforward NN with one hidden layer. $n = 2$, $M = 3$ and $K = 1$.

Furthermore, the technique to introduce an extra input to represent the bias could be used here once again. Therefore, $x_0 = 1$ is defined and Equation 2.14 is then rewritten to

$$a_j = \sum_{i=0}^n w_{ji}^{(1)} x_i. \quad (2.18)$$

So far, the idea behind NNs has been addressed as well as the overall structure. The next challenge is to find values for the parameters and weights introduced in the network that minimize the cost function. A popular method for doing so is SGD [20]. In fact, the authors state that SGD is the most common optimization algorithm in deep learning. SGD uses the gradient to learn weights by using the chain rule [23]. Expressing the gradient mathematically is relatively straightforward, as also shown below. Nevertheless, evaluating the terms can be computationally demanding. The backpropagation algorithm [24] solves this issue by an efficient, iterative and recurrent approach.

An alternative algorithm to SGD is Adam. Adam is also a stochastic gradient based optimizer proposed by Kingma and Ba [25]. The algorithm has proven to be effective and is essentially a combination of the algorithms Adagrad [26] and RMSProp [27]. Adam utilizes a fixed size moving window of past gradients to calculate an exponential moving average, reducing the impact of older gradients step by step. Reddi, Kale and Kumar [28] identified a problem with undesirable convergences for algorithms similar to Adam and proposed a solution where they introduced a long-term memory part to overcome the problem.

In summary, an optimizer utilizes the gradient calculated with the backpropagation algorithm to minimize the objective function. The objective function could be a loss function of any sort and the error is accumulated for every point in the training set [21] as shown by Equation 2.19.

$$L(\mathbf{w}) = \sum_{r=1}^N L_r(\mathbf{w}) \quad (2.19)$$

Furthermore, each activation or input a_j is a weighted sum as described by Equation 2.20.

$$a_j = \sum_i w_{ji} h_i \quad (2.20)$$

The goal now is to derive the loss L_r for weight w_{ji} in the first and second layer. Recall that

the loss is only dependent on the weight through a_j and the hidden unit j . This is where the chain rule is applied, resulting in Equation 2.21.

$$\frac{\partial L_r}{\partial w_{ji}} = \frac{\partial L_r}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \quad (2.21)$$

For the reason of simplicity the following notation in Equation 2.22 is introduced since they will be helpful later. The variable z_i is the input or activation that is sent to the hidden unit j . Recall that the activation function in the case of regression is linear.

$$\begin{aligned} z_i &= \frac{\partial a_j}{\partial w_{ji}} \\ \delta_j &= \frac{\partial L_r}{\partial a_j} \end{aligned} \quad (2.22)$$

Equation 2.21 can now be rewritten with the help of Equation 2.22 to

$$\frac{\partial L_r}{\partial w_{ji}} = \delta_j z_i. \quad (2.23)$$

Now, each δ_j unit is dependent on k units in the next layer which means that the following expression describes how δ_j will be calculated for unit j given the k units in the next layer which results in the formula for backpropagation.

$$\delta_j = \sum_k \frac{\partial L_r}{\partial a_k} \frac{\partial a_k}{\partial a_j} = \phi'(a_j) \sum_k w_{kj} \delta_k \quad (2.24)$$

Thus, the final expression for adjusting the weights in the example from Figure 2.3 with two layers is

$$\begin{aligned} \frac{\partial L_r}{\partial w_{ji}^{(1)}} &= \delta_j x_i \\ \frac{\partial L_r}{\partial w_{kj}^{(2)}} &= \delta_k z_j. \end{aligned} \quad (2.25)$$

Finally, the gradient of the objective function can be efficiently [21] evaluated using the backpropagation algorithm so that the gradient can be used in an optimizer to minimize the loss.

2.4 Recurrent Neural Networks

A RNN is a variant of NN, introducing a neat way of handling time series data and learn long-term dependencies [29]. Variants of RNNs has proven to work well in applications such as speech recognition [30] [31], human action recognition [32] and predicting short-term traffic flow [33]. However, learning long-term dependencies using gradient has been empirically shown to be a difficult task [34]. The gradient either grow or shrink at each time step, causing the gradient to either explode or vanish while propagating several time steps [29]. To facilitate the learning and overcome the problem with the gradient, Hochreiter and Schmidhuber [35] proposed the long short-term memory (LSTM) design back in 1997. In a traditional NN, each input feature has its separate, own parameters so there is no concept of time or memory between the input samples. A RNN on the other hand shares the weights across several time steps [20]. In fact, a RNN can map the entire history of previous input sequences to each output compared to a NN which maps an input to an output without remembering calculations from previous inputs. Remembering calculations refers to the internal state kept

by a RNN which introduces the 'memory' of previous inputs which can affect the output of the network [36]. Figure 2.4 illustrates a standard [36] RNN unrolled in time, meaning that each individual node is associated with a time step [20]. The input vector at time step t is denoted as $\mathbf{x}^{(t)}$ while \mathbf{U} , \mathbf{V} and \mathbf{W} are weight matrices. The hidden state in the network at time t is expressed as $\mathbf{h}^{(t)}$. Similarly, $\mathbf{o}^{(t)}$ is the output at time t . Finally, the bias vector is denoted as \mathbf{b} .

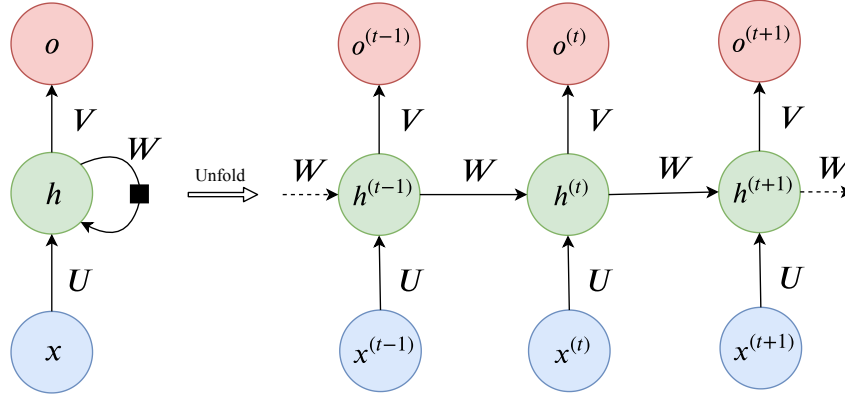


Figure 2.4: An unrolled RNN.

As illustrated in Figure 2.4, the hidden state depends on the current input step as well as the hidden state from the previous time step. This is expressed by

$$\mathbf{a}^{(t)} = \mathbf{U}\mathbf{x}^{(t)} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{b} \quad (2.26)$$

where $\mathbf{a}^{(t)}$ is passed through an activation function f to acquire $\mathbf{h}^{(t)}$ such that

$$\mathbf{h}^{(t)} = f(\mathbf{a}^{(t)}). \quad (2.27)$$

Goodfellow, Bengio and Courville [20] suggest the *hyperbolic tangent* as activation function, but this can of course be any suitable activation function. For instance, Pang et al. [3] utilized the sigmoid function hyperbolic tangent in their model. The sigmoid function is expressed in Equation 2.28 and the hyperbolic tangent in Figure 2.29

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.28)$$

$$f(x) = \tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} \quad (2.29)$$

The output value $\mathbf{o}^{(t)}$ at step t is described by

$$\mathbf{o}^{(t)} = \mathbf{V}\mathbf{h}^{(t)} + \mathbf{c} \quad (2.30)$$

where \mathbf{c} is the bias vector. The output vector could be fed to a softmax function if the task is to output a vector of normalized probabilities or be used directly $\hat{\mathbf{y}}^{(t)} = \mathbf{o}^{(t)}$. An interesting note is that a RNN uses the same weight matrices \mathbf{U} , \mathbf{V} and \mathbf{W} for all time steps. This means that

the difference between two time steps is the input and the hidden state. The same weights are applied which keeps the number of parameters the the model needs to learn low.

The structure of the RNN in Figure 2.4 produces an output at each time step. However, there are alternatives to this approach. For example, the output at time step t could be shared with the hidden state at $t + 1$. The correlation between the number of inputs and outputs can also be adjusted to suite a specific problem. The examples below are divided into three common approaches [20].

- **Many-to-many** Similar to what is illustrated in Figure 2.4. A new input vector is fed at each time step producing a corresponding output.
- **Many-to-one** This approach takes several time steps before an output is produced.
- **One-to-many** One input step results in several outputs.

Training a RNN is similar to training a NN. The loss function $L^{(t)}$ is accumulated across the time steps as described by Equation 2.31.

$$L = \sum_i L^{(i)}(y^{(i)}, \hat{y}^{(i)}) \quad (2.31)$$

Recall that the weights and biases are shared across all time steps so that the gradient has to be calculated for each time step. This algorithm is called *backpropagation through time* (BPTT) [37] and is relatively straight forward to apply in a RNN [20].

2.4.1 Long short-term memory

As previously described, LSTM recurrent networks are special variants of RNNs. LSTMs and gated recurrent units (GRUs) [38] are both approaches for creating sequential models without suffering from an exploding or vanishing gradient [20]. Figure 2.5 is a block diagram of a LSTM unit which replaces the hidden unit $h^{(t)}$ in an ordinary RNN.

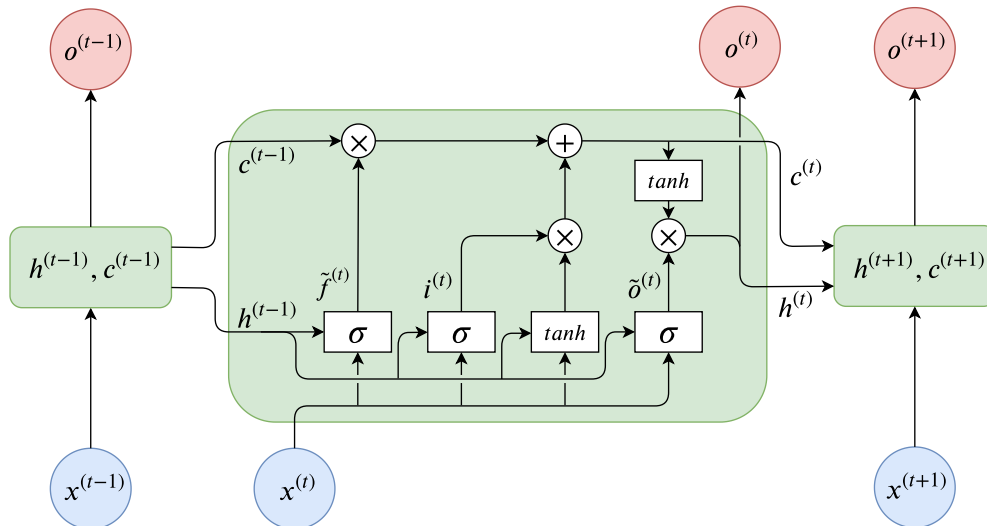


Figure 2.5: Block diagram of a LSTM unit.

Now, for a training set $\chi = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \dots, (\mathbf{x}^{(\tau)}, \mathbf{y}^{(\tau)})\}$ where $t = 1, \dots, \tau$. Let $\mathbf{x}^{(t)}$ and $\mathbf{y}^{(t)}$ be input and targeted output vectors at time t , respectively. The forward propagating equation for the *forget gate* $\tilde{f}_i^{(t)}$ is defined as

$$\tilde{f}_i^{(t)} = \sigma(\mathbf{U}_i^f \mathbf{x}^{(t)} + \mathbf{W}_i^f \mathbf{h}^{(t-1)} + \mathbf{b}_i^f), \quad (2.32)$$

for time step t at LSTM unit i . The subscript i is useful for specifying the LSTM unit in a deep LSTM architecture where several hidden layers are stacked. A good example of deep RNNs is proposed by Graves, Mohamed and Hinton [31], where they used a stack of two hidden layers for speech recognition purposes.

The forget gate was introduced by Gers, Schmidhuber and Cummins [39] as an addition to the standard LSTM architecture with a huge success [20]. The input weight matrix for the forget gate is denoted as \mathbf{U}^f and the weight matrix for the recurrent weights is represented as \mathbf{W}^f . Finally, \mathbf{b}^f is the bias for the forget gate. Equation 2.32 is also visible in Figure 2.5 where it also is described that the input vector $\mathbf{x}^{(t)}$ at time t forms a sum with the hidden layer vector $\mathbf{h}^{(t-1)}$, the bias and of course, the corresponding weights. Furthermore, the forget gate feeds the most important component [20], the *cell state* $c^{(t)}$, with a value between 0 and 1 because of the sigmoid function. The cell state $c^{(t)}$ is a sum of two element wise matrix products calculated as

$$c_i^{(t)} = \tilde{f}_i^t \odot c_i^{(t-1)} + i_i^{(t)} \odot \tanh(\mathbf{U}_i \mathbf{x}^{(t)} + \mathbf{W}_i \mathbf{h}^{(t-1)} + \mathbf{b}_i) \quad (2.33)$$

with the matrices \mathbf{U} and \mathbf{W} as input weights and recurrent weights to the LSTM unit, respectively. The bias to the LSTM cell is denoted by \mathbf{b} , as expected. Note that $i_i^{(t)}$ is another gate named the *input gate*. The input gate is calculated similar to the forget gate but with its own parameters as presented by Equation 2.34.

$$i_i^{(t)} = \sigma(\mathbf{U}_i^i \mathbf{x}^{(t)} + \mathbf{W}_i^i \mathbf{h}^{(t-1)} + \mathbf{b}_i^i) \quad (2.34)$$

The hidden state is propagated through the output gate $\tilde{o}^{(t)}$ which once again utilizes its own parameters \mathbf{U}^o , \mathbf{W}^o and \mathbf{b}^o . The output gate is calculated by

$$\tilde{o}_i^{(t)} = \sigma(\mathbf{U}_i^o \mathbf{x}^{(t)} + \mathbf{W}_i^o \mathbf{h}^{(t-1)} + \mathbf{b}_i^o). \quad (2.35)$$

Thus, the hidden state for time step t is expressed as

$$h_i^{(t)} = \tilde{o}_i^{(t)} \odot \tanh(c_i^{(t)}). \quad (2.36)$$

Finally, the output o^t is calculated by

$$o^{(t)} = \mathbf{V}^o \mathbf{h}^{(t)} + \mathbf{b}^o \quad (2.37)$$

RNNs with LSTM units can take many different forms. For instance, a recent paper authored by Peters et al. [40] suggests that a bidirectional LSTM network performs well in the context of word representation. One reason behind the success is that networks based on LSTM have shown to learn long-term dependencies [35] and challenging sequence processing tasks [41] better than ordinary RNNs [20]. This makes models based on LSTM units an attractive alternative to evaluate and test for problems of similar nature.

2.4.2 Weights and bias initialization

A crucial part in terms of performance in a RNN is to initiate the weight matrices and the bias properly. The approach used by Pang et al. [3] is based on a common idea expressed by Glorot and Bengio [42]. The idea is to draw sampled from a truncated Gaussian distribution centred around zero, where the standard deviation is expressed in Equation 2.38, where N_{in} and N_{out} is number of input and output units to the neuron, respectively.

$$\sigma = \sqrt{\frac{2}{N_{in} + N_{out}}} \quad (2.38)$$

Similarly, He et al. [43] proposes an idea with a slightly different standard deviation described in Equation 2.39.

$$\sigma = \sqrt{\frac{2}{N_{in}}} \quad (2.39)$$

2.5 Regularization

Recall that a core problem in machine learning is to make the models perform well on previously unseen data. Regularization refers to modifications made to the learning algorithm with the intention to reduce the generalization error. [20].

A common strategy to avoid overfitting is to apply *early stopping*. Early stopping is a technique where the training and test error is observed during training. In the normal case, both errors initially decreases during the training phase. However, instead of iterating the training data until the error on the training data has converged, early stopping stops the iteration when the observed test error has worsened for the last p times. The parameter with the lowest test error is kept and return by the time early stopping kicks in instead of returning the parameters that happened to be there at the end of the iterations.

Another powerful regularization method is *dropout* which has been successfully applied to LSTM units in two ways by Gal and Ghahramani [44]. Dropout was proposed as a technique for reducing overfitting by Srivastava et al. [45] by randomly dropping out units in the network. This breaks co-adaptions in the network during the training phase since no unit is taken for granted. Another point of view is to think that dropout trains several models consisting of sub networks [20]. One way to apply dropout to LSTM networks as described by Gal and Ghahramani, is to apply it to the inputs and/or the outputs. The other approach is to apply dropout to the connection between the LSTM units, meaning that connections between time steps is randomly broken.

2.6 Gaussian processes

Another regression approach with promising results [46] [47] is based on a Gaussian processes (GPs) [48], which is a Bayesian non-parametric approach to model distributions over functions. An appealing feature of GPs is the fact that they naturally capture uncertainty of the model since they produce a distribution for the targeted value rather than a solitary value which a standard NN does, for instance. Uncertainty could however be captured in NN type models by utilizing dropout as a Bayesian approximation [49]. Rasmussen [48] describes how a function f is distributed as a GP where f constitutes of a mean function $m(x)$ and a covariance function $k(x, x')$ such that

$$f \sim GP(m(\cdot), k(\cdot, \cdot)). \quad (2.40)$$

Given that there is enough prior information on a given dataset the process of choosing suit-

able functions $m(x)$ and $k(x, x')$ is straight forward. This is not the typical case in machine learning in general or GP regression in particular which means that there must be a way of choosing suitable functions. This choosing process is referred to as training the GP model. However, if there is a situation where prior knowledge is present it is convenient to code in that knowledge by picking a covariance function that matches the needs [48]. The prior knowledge is typically encoded in the structure of the kernels by for instance, combining different kernels by a sum or a multiplication. There are still parameters to learn even when domain knowledge is present.

Rasmussen [48] describes some common covariance functions and their attributes. It is interesting to consider a few of them to get an idea of how they work. For instance, the squared exponential covariance function is defined as

$$k(x, x') = \exp\left(-\frac{|x - x'|^2}{2\ell^2}\right), \quad (2.41)$$

where ℓ is the *characteristic length-scale* parameter of the GP, adjusting when two points x and x' influences each other. A GP using the squared exponential covariance function is very smooth since the covariance function has mean squared derivatives of all orders. Furthermore, yet another covariance function described by Rasmussen [48] is the rational quadratic which is defined by Equation 2.42 with $\alpha, \ell > 0$.

$$k(x, x') = \left(1 + \frac{|x - x'|^2}{2\alpha\ell^2}\right)^{-\alpha} \quad (2.42)$$

The rational quadratic covariance function can be views as an abstraction of the squared exponential function in a way that the rational quadratic add together many different squared exponential functions of different length ℓ . In fact, infinite many [48].

The two covariance functions presented here is perhaps a starting point for discovering more sophisticated functions with other attributes. Some which are periodic, locally periodic, linear or constant. There are of course more details to explore here where kernel also could be combined by for instance addition or multiplication. However, the purpose of this section is to introduce a highly feasible alternative approach for solving regression problems.

2.7 Evaluation techniques

In order to evaluate how regression models perform, some evaluation techniques are needed. One interesting aspect to look at is how far away the predictions are from the true targets, on average. Let the i -th observed value be denoted by y_i , and the corresponding predicted value by \hat{y}_i . The mean squared error (MSE) is then defined as

$$MSE(y, \hat{y}) = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n} \quad (2.43)$$

where n is the number of samples. Commonly, the root mean square error is used, which expressed by

$$RMSE(y, \hat{y}) = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}} = \sqrt{MSE(y, \hat{y})}. \quad (2.44)$$

Willmott and Matsuura [50] suggested that RMSE should not be used because of the inappropriate representation of the average error. Instead, they suggest that the mean absolute error (MAE), calculated with equation 2.45, is a more natural representation of the average error. However, Chai and Draxler [51] showed that RMSE can be more suitable than MAE

in, for instance, the case where the error distribution is expected to be Gaussian. They also proposed that a combination of several metrics, including RMSE and MAE should be used.

$$MAE(y, \hat{y}) = \frac{\sum_{i=1}^n |y_i - \hat{y}_i|}{n} \quad (2.45)$$

Furthermore, the mean absolute percentage error (MAPE), defined as

$$MAPE(y, \hat{y}) = \frac{\sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|}{n} * 100\% \quad (2.46)$$

is yet another suitable metric for evaluating the quality of regression models [52].

3 Data

This chapter introduces the data source used in this thesis. A good understanding of the possibilities and limitations of the data is essential for conducting good analyzes.

Currently, ÖT has a system that collects AVL data, which essentially contain GPS information about the position of the vehicle as described in chapter 2. This data is processed and the resulting information is provided through the *Nordic Public Transport Interface Standard* (NOPTIS) [53]. This information is then used by other local systems such as travel planners. It is also used by the organization Trafiklab [54], which transform part of the available information into the *General Transit Feed Specification* (GTFS) and GTFS Realtime (GTFS-RT) [55] formats. They provide application programming interfaces (APIs) for a set of public transport agencies in Sweden, including ÖT. Figure 3.1 depicts an overview of how the data flow works. The data is transmitted with the internet protocol UDP, suggesting that some data might get lost due to interference in the transmission.

Vehicle positions are updated approximately every second. Trip updates and service alerts are updated around every 15 seconds, while the GTFS feed is updated on a daily basis. Every update is a full version of the current state including the location of all vehicles and status of all trips being active at the very moment of the update.

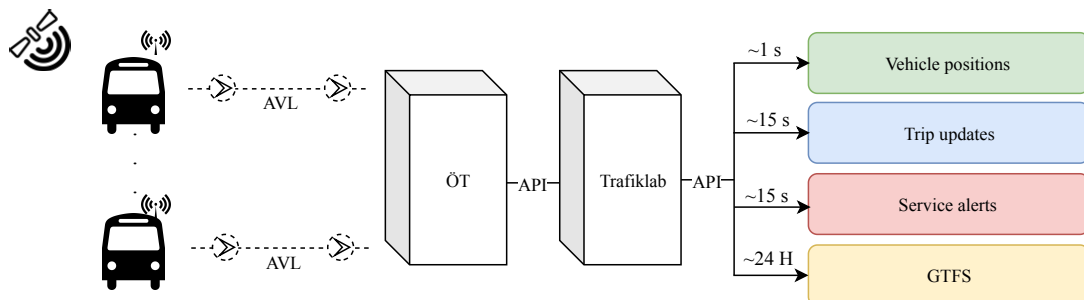


Figure 3.1: Overview of the data flow. Each bus is equipped with a GPS transmitter estimating the current position. This information along with calculated arrival, departure and prognosis-information from ÖT is automatically propagated to Trafiklab, which in turn make the information available through a set of APIs.

3.1 GTFS

The GTFS format is a format developed by Google [56] and defines a way to express public transport information such as schedules and geographic information. A GTFS feed consists of a set of text files, where each text file encapsulate pieces of information separated by commas. The following text files are available at Trafiklab:

- agency.txt
- calendar.txt
- calendar_dates.txt
- feed_info.txt
- routes.txt
- shapes.txt
- stops.txt
- stop_times.txt
- transfers.txt
- trips.txt

Figure 3.2 is a simplified schema over the relation between the individual files in the GTFS feed. Foreign keys correspond to id's that are used to identify an entry in a separate file.

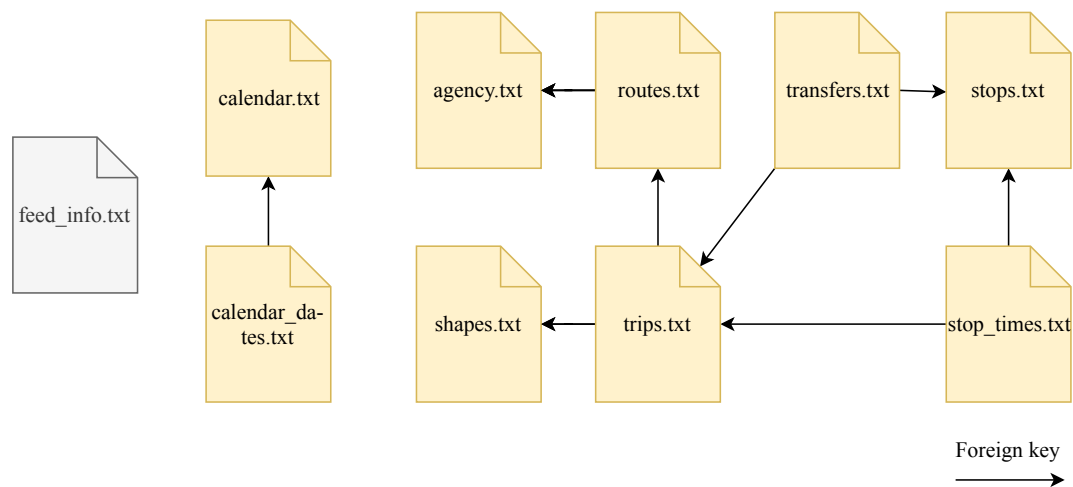


Figure 3.2: Simplified schema of the relations between individual files in the GTFS feed available at Trafiklab.

The file `feed_info.txt` contains information about the GTFS feed itself and can be used to determine whether the feed has been updated or not. Thus, data can conveniently be request only after an update has been made to the feed.

3.2 GTFS Realtime

In addition to the static traffic information provided by GTFS, GTFS-RT is an extension for communicating traffic information in realtime. GTFS-RT consists of three main components and supports information flows about trip updates, vehicle positions and service alerts.

Trip updates contain information regarding delays, changes in routes and cancellations. The delay parameter can be speculative for future stops, i.e. a prediction, or represent the difference between the scheduled and actual arrival time for past stops. Vehicle positions on the other hand contain the GPS coordinate information from the GPS devices onboard the buses. Service alerts are used when there are disruptions in the traffic system. A traffic controller can create service alerts messages with textual information about the problem. Ordinary delays or cancellations is communicated through trip updates.

The GTFS-RT feed from ÖT is available through Trafiklab [54] via the *protocol buffer* [57] file format which aims to be smaller, simpler and faster than ordinary XML. The data used in this thesis was collected using the protocol buffer format and a third party library [58] for reasons of performance and to facilitate the interpretation of the file content. Later on, the content of the files were organized into a document oriented database. The database sorted the information by trip and date which facilitates information retrieval since the data of a trip is no longer spread out over a large number of files.

3.3 Stops

The geographic information of the stop locations can be extracted from the static GTFS feed, and Figure 3.3 illustrates all stop locations that ÖT operates. A stop is divided into three categories. An ordinary stop, a station or a station entrance or exit. A station can consist of one or more ordinary stops, and the station is a parent to the stops that belongs to the station.

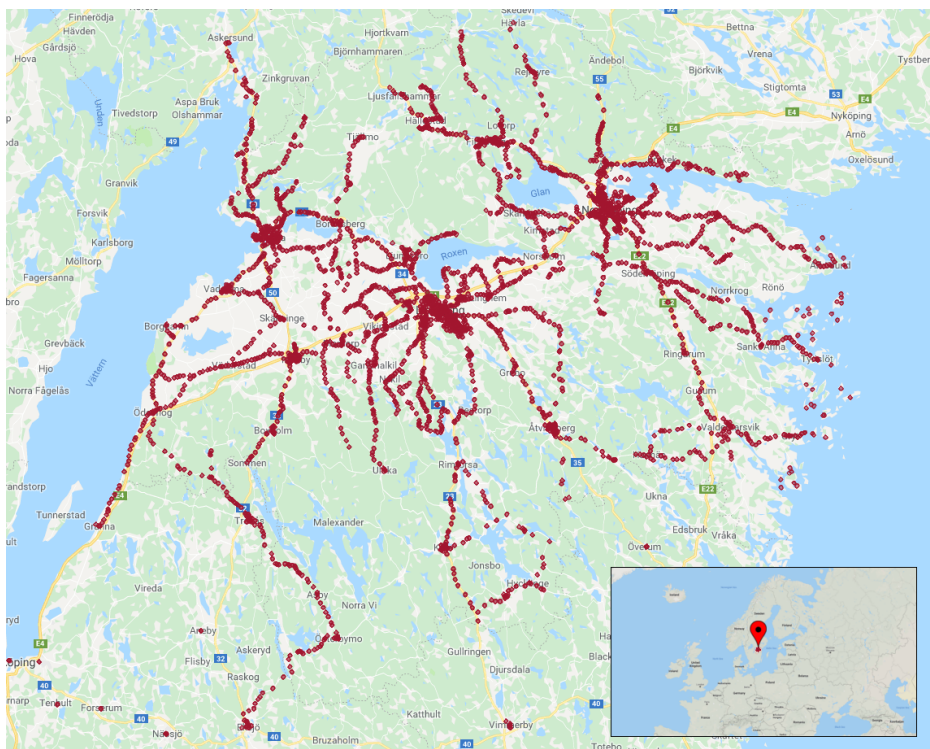


Figure 3.3: Illustration over the bus stop locations. Each red circle corresponds to a geographic location labelled as a type of stop. A stop in this figure is included in or several means of transport, including bus, train, tram, or boat.

3.3.1 Timed stops

Recall that timed stops are stops for which the bus driver can not depart earlier than scheduled. They are stops of certain importance to keep a good traffic flow. The bus driver has to wait for the departure time if the bus has arrived at a timed stop earlier than the timetable indicates. It is therefore interesting to consider them in an analyzes since they perhaps can explain certain patterns in the data. For instance, they might be the reason why buses arriving early end up in the same phase, complying with the timetable at the same time. Also, arriving early to a timed stop does not necessarily have dull consequences for passengers-to-be if the driver complies with the regulations while some passengers can be pleased about arriving a bit earlier than planned if that particular timed stop is their final stop.

3.4 Trips

A *trip* in this thesis refers to a concrete instance of a bus journey from the first bus stop to the very last one. More formally, a *trip* is a sequence of two or more stops ($s = 1, \dots, S$), that follows the same predefined schedule. The arrival and departure times for the sequence of S stops is denoted by the vectors defined in Equation 3.1a and 3.1b, respectively.

$$\mathbf{a} = (a_2, \dots, a_S) \quad (3.1a)$$

$$\mathbf{d} = (d_1, \dots, d_{S-1}) \quad (3.1b)$$

Thus, $trip = [\mathbf{a}, \mathbf{d}, S]$ which is a triple with arrival and departure times for stops on the *trip*. The first stop on a trip is only associated with a departure time because of the relevance to the passengers. Similarly, only the departure time for the stop $S - 1$ is considered.

Figure 3.4 illustrates the planned trajectory of a typical trip. This particular trip goes from a major city to another one in the dataset, namely from Linköping to Norrköping. The trip visits seven stops in the first city, then goes along a main road to reach the remaining seven stops in the second city.

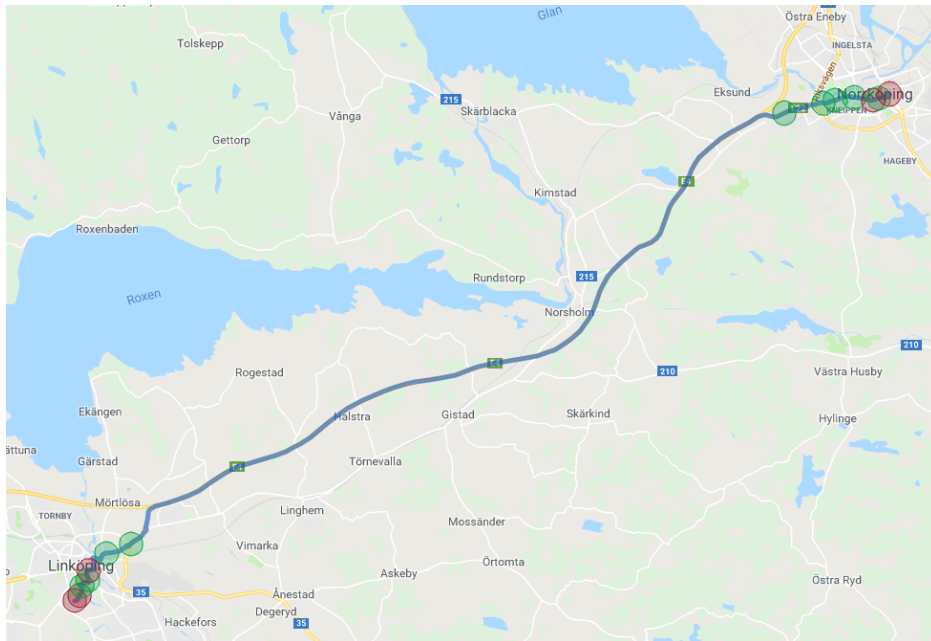


Figure 3.4: Visualization of a trip. The blue line represents the planned trajectory. The red and green circles are stops on the trip where red circles represents timed stops.

So far, the static properties of trips have been investigated. The dataset contains information about the actual arrival and departure times for all trips. As previously described, the arrival and departure times at stops are continuously updated in the trip updates feed. A trip update with the uncertainty parameter set to 0 for a stop is considered to be the observation for when the bus arrived and left the stop. When the uncertainty parameter is missing, the stated arrival time value is a speculation produced by a system of when the bus should arrive. The bus driving is further investigated to get an idea of how good the schedule and observed outcomes of a trip align. The schedule is as it should be, and it is evaluated how well the driving does in-fact match the schedule. There are probably parts of trips where the schedule for instance, matches the observed arrival times well most of the time. In such a situation, the schedule alone works just fine for predicting the arrival time. On the other hand, there are probably also parts of trips where the observations are quite far from the schedule, indicating room for improvements. With this in mind, the MSE for arrival and departure times is utilized as described in Equation 3.2a and 3.2b, respectively. N is the number of times the trip has been driven. a_s^i represents the arrival time for stop s on the i -th time the trip was driven. Furthermore, \hat{a}_s^i is a prediction of some sort on that particular stop. This prediction could be based on human experience, the schedule, or as in this first case, the real world observation. The same principles apply to the departure times as well.

$$MSE_{arrival}(a, \hat{a}) = \frac{\sum_{i=1}^N \sum_{s=2}^S (a_s^i - \hat{a}_s^i)^2}{(S-1)N} \quad (3.2a)$$

$$MSE_{departure}(d, \hat{d}) = \frac{\sum_{i=1}^N \sum_{s=1}^{S-1} (d_s^i - \hat{d}_s^i)^2}{(S-1)N} \quad (3.2b)$$

A concrete example from the data is illustrated in Figure 3.5, which shows an instance of a trip following the stop sequence and trajectory in Figure 3.4. On this particular day, the bus departed around a minute after schedule from the first stop on the trip and arrived a few minutes early at the last one. The one-minute deviation from the schedule is marked since one minute is considered to be an acceptable deviation from the schedule for stops that are not labelled as timed stops. Therefore, passengers has to be aware that a bus can departure up to one minute before the schedule for bus stops that are not considered to be timed stops.

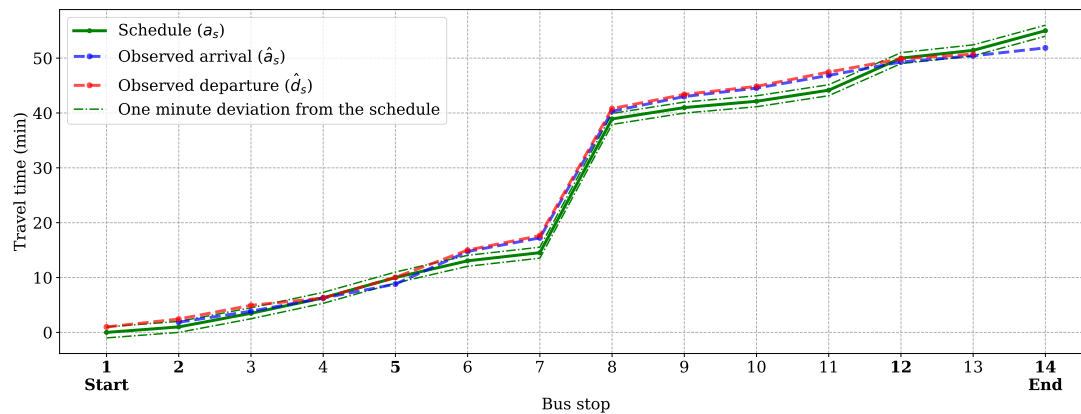


Figure 3.5: Observed arrival and departure times compared to the schedule for a trip from Linköping to Norrköping, 16.00 local time 2019-02-18. Bold bus stops are timed stops.

To get one more perspective on how well a driven trip matches the schedule including the acceptable deviation, a modification to the cost functions denoted with (*) is made. This modification yields an error of zero if the prediction appears to be within the acceptable region. On the other hand, if the prediction is more than one minute wrong, the mean squared error

function starts kicking in and gives an error similar to the ordinary MSE function. Equation 3.3 shows how MSE^* is calculated.

$$MSE_{arrival}^*(a, \hat{a}) = \begin{cases} \frac{\sum_{i=1}^N \sum_{s=2}^S (a_s - \hat{a}_s)^2 - 1}{(S-1)N}, & \text{if } |a_s - \hat{a}_s| > 1 \\ 0, & \text{otherwise} \end{cases} \quad (3.3a)$$

$$MSE_{departure}^*(d, \hat{d}) = \begin{cases} \frac{\sum_{i=1}^N \sum_{s=1}^{S-1} (d_s - \hat{d}_s)^2 - 1}{(S-1)N}, & \text{if } |d_s - \hat{d}_s| > 1 \\ 0, & \text{otherwise} \end{cases} \quad (3.3b)$$

Recall that the schedule is being predicted by the observed outcome in this first perspective. Applying the MSE metrics in Equation 3.2 and the MSE^* metrics in Equation 3.3 yields the results in Table 3.1.

Trip		MSE	MSE*
Linköping - Norrköping 16.00 local time 2019-02-18 (N = 1, S = 14)	arrival	3.30	2.50
	departure	3.45	2.68

Table 3.1: MSE and MSE* for the example in Figure 3.5.

Furthermore, the observed arrival and departure times for the same trip on a different day varies as anticipated. For instance, Figure 3.6 illustrates the observed time values for the same trip as Figure 3.5 on a different date. This time, the bus departed a bit early from the first stop and experienced some kind of delay between the fifth and sixth stop, causing the bus to experience a delay over the one minute mark for the rest of the journey.

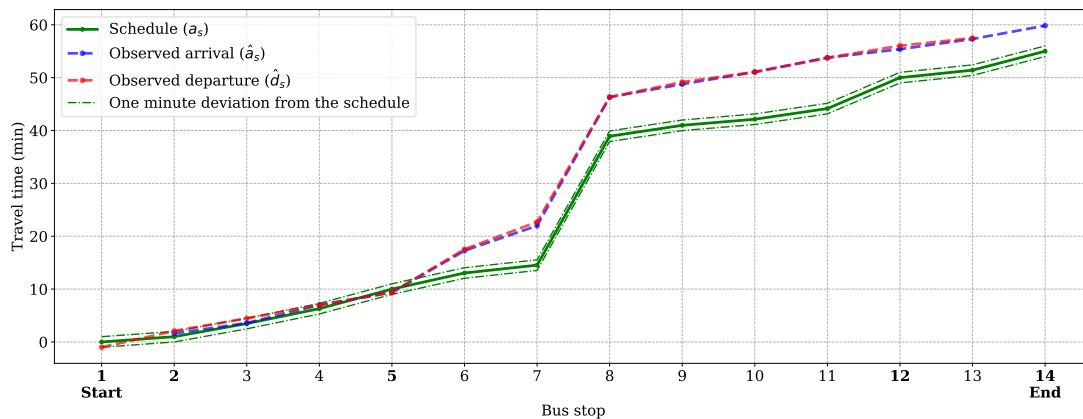


Figure 3.6: Observed arrival and departure times compared to the schedule for a trip from Linköping to Norrköping, 16.00 local time 2019-02-26.

The delay seen in Figure 3.6 gives, as expected, a greater error in the cost functions as shown in Table 3.2.

Trip		MSE	MSE*
Linköping to Norrköping 16.00 local time 2019-02-26 (N = 1, S = 14)	arrival	34.56	33.77
	departure	34.90	34.05

Table 3.2: MSE and MSE* for the example trip in Figure 3.6.

The cost functions gives an error for the whole trip and it can be difficult to identify what causes the numbers without any further information. To give a more detailed view of how the

error develops and where, Figure 3.7 illustrates the MSE for the two example trip instances. The figure confirms the measured values in Table 3.1 and Table 3.2 by showing that the error was much greater on the date 2019-02-26. It also reveals that the departure error usually is greater, indicating that the bus mostly was late. One exception is bus stop number five in Figure 3.7a where the arrival error is greater than the departure error. On this particular day, the driver complied with the requirements for timed stops and awaited the departure time, which can be concluded since the departure error from the same stop is zero.

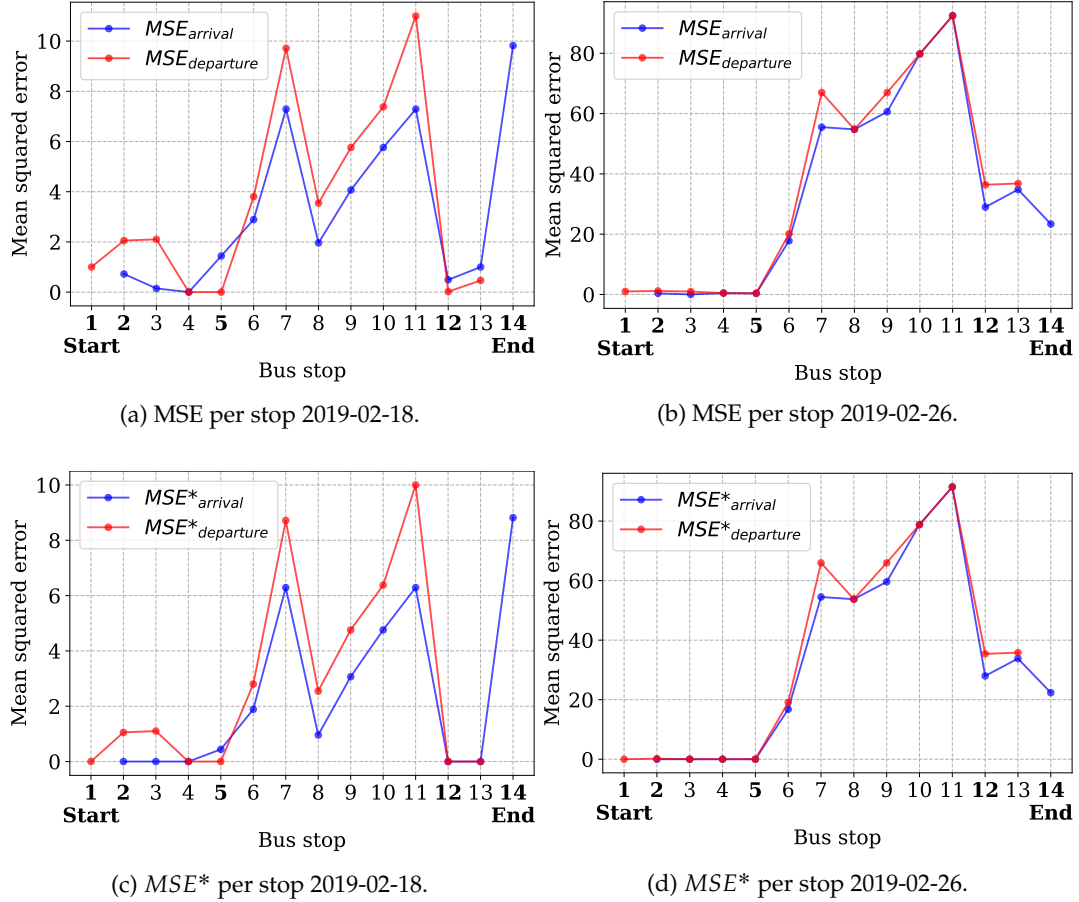


Figure 3.7: MSE and MSE* for the trip from Linköping to Norrköping 16.00 local time at two different dates.

So far, the only perspective considered is how well the observed arrival and departure times predict the schedule by using MSE and MSE^* for a single day. Before considering the performance for a trip based on several days, i.e. $N > 1$, a couple of complementary cost functions are defined. The first one is MAE , which is calculated as described in Equation 3.4.

$$MAE_{arrival}(a, \hat{a}) = \frac{\sum_{i=1}^N \sum_{s=2}^S |a_s^i - \hat{a}_s^i|}{(S-1)N} \quad (3.4a)$$

$$MAE_{departure}(d, \hat{d}) = \frac{\sum_{i=1}^N \sum_{s=1}^{S-1} |d_s^i - \hat{d}_s^i|}{(S-1)N} \quad (3.4b)$$

Once again a variant to the cost function is introduced with the purpose of taking the allowed

one minute deviation from the schedule into account. This time the cost function is denoted by MAE^* and is defined in Equation 3.5.

$$MAE_{arrival}^*(a, \hat{a}) = \begin{cases} \frac{\sum_{i=1}^N \sum_{s=2}^S |a_s - \hat{a}_s| - 1}{(S-1)N}, & \text{if } |a_s - \hat{a}_s| > 1 \\ 0, & \text{otherwise} \end{cases} \quad (3.5a)$$

$$MAE_{departure}^*(d, \hat{d}) = \begin{cases} \frac{\sum_{i=1}^N \sum_{s=1}^{S-1} |d_s - \hat{d}_s| - 1}{(S-1)N}, & \text{if } |d_s - \hat{d}_s| > 1 \\ 0, & \text{otherwise} \end{cases} \quad (3.5b)$$

Finally, the $RMSE$ is just the square root of the MSE defined by Equation 3.6.

$$RMSE_{arrival}(a, \hat{a}) = \sqrt{MSE_{arrival}(a, \hat{a})} \quad (3.6a)$$

$$RMSE_{departure}(d, \hat{d}) = \sqrt{MSE_{departure}(d, \hat{d})} \quad (3.6b)$$

For the same reason as for the previously defined metrics, $RMSE^*$ is introduced and calculated as in Equation 3.7.

$$RMSE_{arrival}^*(a, \hat{a}) = \sqrt{MSE_{arrival}^*(a, \hat{a})} \quad (3.7a)$$

$$RMSE_{departure}^*(d, \hat{d}) = \sqrt{MSE_{departure}^*(d, \hat{d})} \quad (3.7b)$$

Now, the next interesting step is to consider how the observed arrival and departure times predict the schedule over a set of different dates to be able to distinguish any patterns. Figure 3.8 illustrates the observed arrival and departure times for the example trip in the period 2019-02-11 to 2019-03-04. The corresponding MSE per stop for all occurrences in the dataset is depicted by Figure 3.9

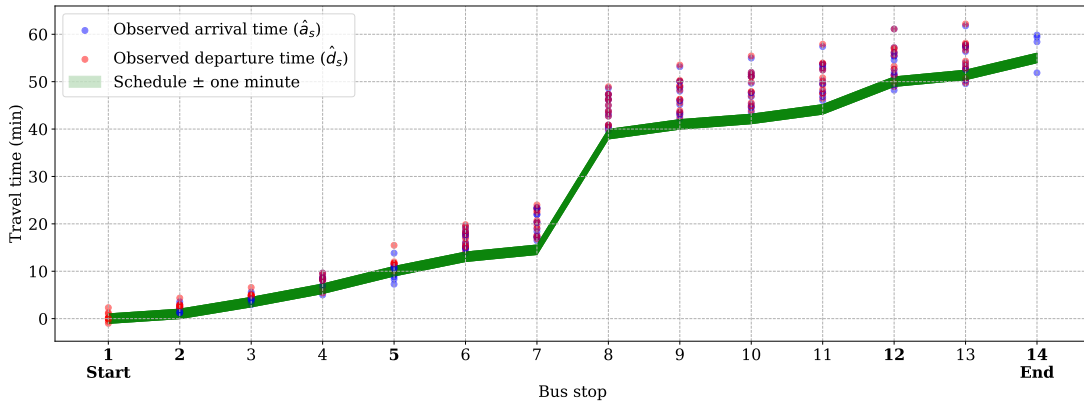


Figure 3.8: Observed arrival and departure times compared to the schedule for a trip from Linköping to Norrköping, 16.00 local time during the period 2019-02-11 to 2019-03-04, $N = 13$.

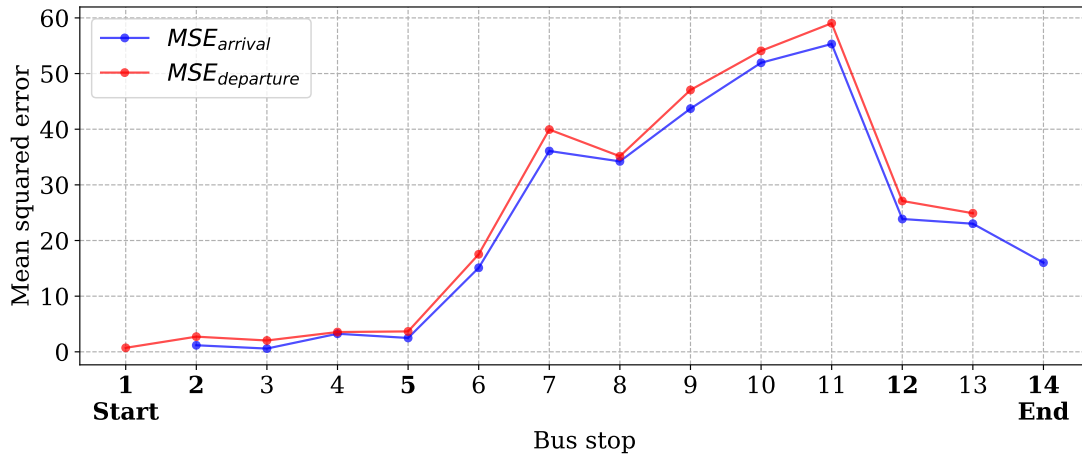


Figure 3.9: MSE for a trip from Linköping to Norrköping, 16.00 local time during the period 2019-02-11 to 2019-03-04, $N = 13$.

To make the example complete and to get an idea of how good the schedule can be predicted with the observations based on the cost functions defined thus far, Table 3.3 shows the results of applying the error functions.

Trip		MSE	MSE*	MAE	MAE*	RMSE	RMSE*
Linköping to Norrköping 16.00 local time ($N=13, S=14$)	arrival	23.96 ± 21.92	23.12 ± 21.87	3.73 ± 1.95	2.84 ± 1.91	4.36 ± 2.23	4.22 ± 2.30
	departure	24.19 ± 21.90	23.34 ± 21.84	3.74 ± 1.95	2.85 ± 1.89	4.39 ± 2.22	4.26 ± 2.29

Table 3.3: Error function results for the example trip with $N = 13$ and $S = 14$.

A set of basic cost functions is now defined. There are positional observations on approximately 6320 bus trips and 156 routes in the dataset. Thus, there are many situations and trips to investigate more closely. In this thesis, a set of routes out of these 156 were hand-picked to reduce the scope and enable fruitful, more in-depth analyzes. A set of routes with slightly different characteristics have been selected to include different types of routes. For instance, the selected routes ranges from relatively short city-routes with a few stops to routes which are among the longest in the dataset with many stops. The trips belonging to a selected routes will be included in the continued analysis.

3.5 Routes

A route is a set of one or several trips that appears to the passengers as one unit identified with a route number. Each trip belongs to exactly one route. Let a route R be a set of T trips such that $R = \{trip_1, \dots, trip_T\}$. The dataset contains 156 such bus routes. Table 3.4 shows more information about the whole dataset available in this thesis as well as a comparison to the dataset used by Pang et al. [3]. Note that approximately 95% of their routes are located in central Beijing, which appears to be a more busy area with more trips per route compared to the dataset in this thesis. On the other hand, the dataset in this thesis includes a larger number of routes. The routes also has more varying lengths and stop counts to be considered.

Dataset	Routes	Route length (km)			Trips per route			Stops per route		
		Min	Max	Mean $\pm \sigma$	Min	Max	Mean $\pm \sigma$	Min	Max	Mean $\pm \sigma$
Pang et al. [3]	47	5.42	24.58	12.02 ± 4.92	718	3009	1606.53 ± 527.62	6	39	18.08 ± 6.28
This thesis	156	0.96	81.89	23.92 ± 15.66	1	307	40.56 ± 54.76	2	60	23.76 ± 13.50

Table 3.4: Comparison of characteristics between two datasets.

3.6 Baselines

A new model proposal must be comparable to existing ways of predicting the arrival in order to be able to reason about the quality of the predictions. There are currently two reasonable types of predictions available in the dataset. The first one is to predict the arrival times by using the scheduled time for each individual stop. That means that the only information used to predict arrival times is the schedule without taking the current state or historical events into account. Let this method, based on readings from the timetable be denoted as *Timetable Based Predictions* (TBP). Secondly, the dataset contains arrival time predictions produced by a prediction system. The implementation of that particular system is not publicly available. However, the system is mainly based on statistical models on historical data and it is currently used in production. Let this second method be denoted as the *Current Prediction System* (CPS).

The CPS is associated with stops along the trips. The current configuration starts with all the remaining stops getting a predicted arrival time from the first stop. However, the CPS can be configured to start predicting arrival times actively already from the first stop, but this is not the case today. When the bus arrives at the next station, the actual arrival time is observed and the CPS produces a new set of arrival time predictions for the remaining stops. Figure 3.10 illustrates how the CPS works when the bus travels along a trip.

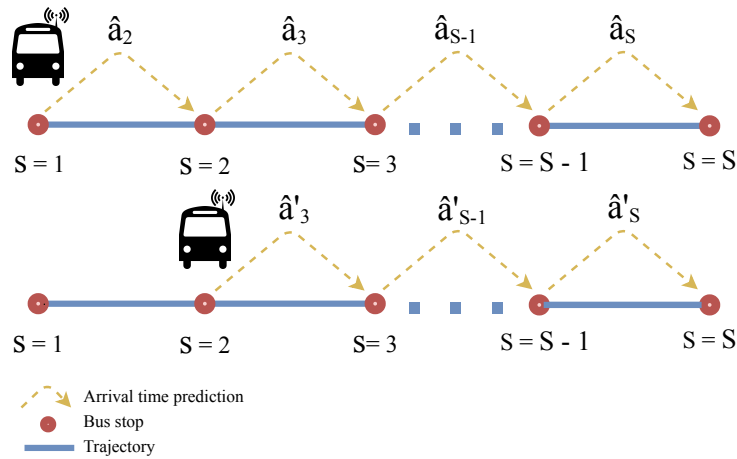


Figure 3.10: Illustration of how the CPS operates.

In order to capture the quality of the predictions, from the first stop to the second last one, the cost functions defined earlier needs to be modified slightly. Let Δ be the set of remaining stops, such that $\Delta = \{1, \dots, S - s\}$. Thus, $\hat{a}_{s,s+\Delta}$ is the predicted arrival time at stop s , for a set Δ of ahead stops. Furthermore, the MSE for evaluating predictions is therefore defined as in Equation 3.8.

Note that these values are calculated per route, and the number of trips on a route is varying depending on how many trips the route consists of.

$$MSE(a, \hat{a}) = \frac{\sum_{i=1}^N \sum_{s=1}^{S-1} \frac{1}{S-s} \sum_{\Delta=1}^{S-s} (a_{s+\Delta}^i - \hat{a}_{s+\Delta}^i)^2}{(S-1)N} \quad (3.8)$$

Similarly, the RMSE is defined in Equation 3.9.

$$RMSE(a, \hat{a}) = \frac{\sum_{i=1}^N \sum_{s=1}^{S-1} \sqrt{\frac{1}{S-s} \sum_{\Delta=1}^{S-s} (a_{s+\Delta}^i - \hat{a}_{s+\Delta}^i)^2}}{(S-1)N} \quad (3.9)$$

Finally, the MAE is defined by Equation 3.11.

$$MAE(a, \hat{a}) = \frac{\sum_{i=1}^N \sum_{s=1}^{S-1} \sum_{\Delta=1}^{S-s} |a_{s+\Delta}^i - \hat{a}_{s+\Delta}^i|}{(S-1)NS} \quad (3.10)$$

Another useful perspective is introduced by the metric *MAPE*, which describes the mean absolute percentage error. This metric is defined as

$$MAPE(a, \hat{a}) = \frac{\sum_{i=1}^N \sum_{s=1}^{S-1} \sum_{\Delta=1}^{S-s} \left| \frac{a_{s+\Delta}^i - \hat{a}_{s+\Delta}^i}{a_{s+\Delta}^i} \right|}{(S-1)NS} * 100. \quad (3.11)$$

The baseline results can be found in Chapter 5.



4 Method

Being able to replicate a study is perhaps one of the more important aspects in order to make the conducted work relevant for others. This chapter describes the method used in this thesis with the aim to be as precise and clear as possible.

4.1 Data selection

In the work by Pang et al. [3] one LSTM model was trained for each route. That approach was followed in this thesis. Thus, a set of routes were selected to build models on for practical reasons. The goal was to construct a set of routes with different characteristics such as length, number of stops and type of route. Some routes in the large dataset are city routes while some others are driven on main road between cities or even on minor roads. The data used in this thesis was collected during the dates 2019-02-11 to 2019-03-29. Figure 4.1 depicts the trajectories of the routes that were selected for further investigation. Table 4.1 gives more details about the selected routes one by one, including three route types. A city route type is considered to be a route where a majority of the trajectory is located in densely populated areas. Inter-city routes are driven between two densely populated areas. Finally, inter-city express routes are also driven between two densely populated areas but with fewer stops in order to reduce travel times.

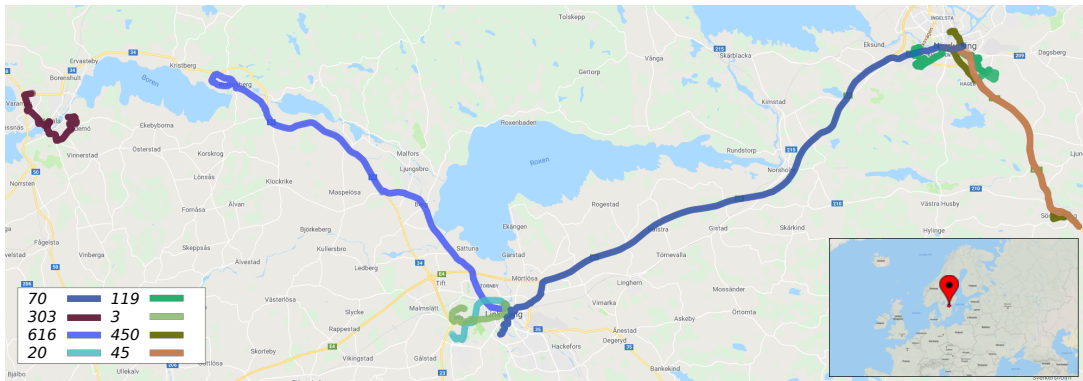


Figure 4.1: Trajectory for the routes for which LSTM models were developed.

Route number	Route length (km)	Trips per route	Stops per route	Route type
70	44.26	32	14	Inter-city express
303	10.07	63	25	City
616	32.97	50	32	Inter-city
20	6.59	20	6	City
119	13.59	191	29	City
3	7.15	307	17	City
450	21.73	40	28	Inter-city
45	18.76	43	8	Inter-city express

Table 4.1: Hard measures on the selected routes from the large dataset.

4.2 Loss functions

The loss function is responsible for giving feedback in terms of cost so that the weights can be updated to reduce the cost in the next iteration. In other words, the loss function controls what the model should strive for in terms of predictions. This idea was used to construct two customized loss functions. One for penalizing underestimations and another for penalizing overestimations. This refers to the two use-cases for passengers and passengers-to-be. Recall that an overestimated arrival time for passengers-to-be is much worse than an underestimated time since it can cause passengers-to-be to be left behind. Similarly, underestimated arrival times for passengers already on the bus can be inconvenient if it is of importance to be on time. Underestimated arrival times in such situations makes it more difficult to plan ahead. Passengers who expected to be there at the given time to, for example, catch up with another bus or attend a meeting are probably disappointed. Hence, there are situations when overestimations and underestimations are unwanted and should be avoided. Chen, Keng and Moreno [59] proposed a solution for their dataset when they wanted to penalize overestimations more heavily. One LSTM network architecture was trained per route. In the case where underestimations and overestimations are to be penalized, the loss is represented by the Equations in 4.1 and 4.3, respectively. Note that the coefficients $\frac{1}{3}$ and $\frac{1}{6}$ were empirically found to be suitable for this domain by evaluating the distribution of errors during the design phase. They are interesting design parameters and subjects to optimization and customization in order to achieve the desired goal. The following loss function defines U

$$\mathcal{L}_U(y_{s+\Delta}, \hat{y}_{s+\Delta}) = \sum_{s=1}^{S-1} \sum_{\Delta=1}^{S-s} \text{loss}(y_{s+\Delta} - \hat{y}_{s+\Delta}) \quad (4.1)$$

where

$$\text{loss}(x) = \begin{cases} e^{-\frac{x}{3}} - 1, & \text{if } x < 0 \\ e^{\frac{x}{6}} - 1, & \text{otherwise} \end{cases} \quad (4.2)$$

Similarly, O is defined by

$$\mathcal{L}_O(y_{s+\Delta}, \hat{y}_{s+\Delta}) = \sum_{s=1}^{S-1} \sum_{\Delta=1}^{S-s} \text{loss}(y_{s+\Delta} - \hat{y}_{s+\Delta}) \quad (4.3)$$

where

$$\text{loss}(x) = \begin{cases} e^{-\frac{x}{6}} - 1, & \text{if } x < 0 \\ e^{\frac{x}{3}} - 1, & \text{otherwise} \end{cases} \quad (4.4)$$

The notations U and O represents the loss functions where the errors are desired to be biased towards overestimations and underestimation, respectively. It turned out empirically that smaller values could be used so that erroneous predictions is more heavily penalized. However, the loss became large and the function may need to be relaxed for even longer sequences

to prevent the loss from exploding. However, the principle that one side is more penalized should remain the same.

Another way of calculating the loss, as suggested by Pang et al. [3], is to use the absolute value together with a smoothing around the origin. Their loss function is described by Equation 4.6 and it was also used in this thesis.

$$\mathcal{L}_{\text{Pang}}(y_{s+\Delta}, \hat{y}_{s+\Delta}) = \sum_{s=1}^{S-1} \sum_{\Delta=1}^{S-s} \text{loss}(y_{s+\Delta} - \hat{y}_{s+\Delta}) \quad (4.5)$$

where

$$\text{loss}(x) = \begin{cases} \frac{x^2}{2}, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases} \quad (4.6)$$

As the authors also describe, their loss function is less sensitive to outliers than for instance O and U because of the wider span.

Another loss function considered, perhaps more commonly used for regression problems, is the MSE. The purpose of including the MSE in the set of loss functions is largely to be used as a reference to the other functions. Finally, the last loss function investigated was MAE^* . This function builds upon the loss proposed by Pang et al. [3] with resilience to outliers as well as the assumption that one minute error is an accepted range. Figure 4.2 shows the loss functions that were further investigated.

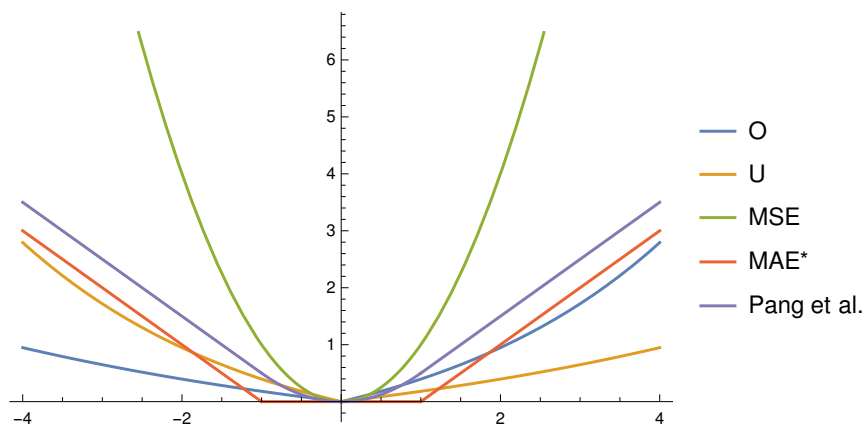


Figure 4.2: A plot of the loss functions examined.

4.3 Models

The selection of model architectures is influenced by the work of Pang et al. [3], which used a two stacked LSTM network with an input vector with eight features. The reason for following their approach is that it is tested with good results on a different dataset in a similar domain and it limits the number of possible constellations. After all, there might be a various set of architectures giving interesting results and it is outside the scope of this thesis to dive deeply into specific design choices. The focus will instead be directed towards different loss functions and their effect on the predictions. Recall that the predictions are made from two perspectives. Passengers already on the bus and passengers yet to board with their respective needs. With that in mind, five different loss functions are proposed. The network architecture is the same and the loss function is the only difference between the trained models. Table 4.2 shows the loss functions evaluated in this thesis. Recall that the loss functions MSE and $\mathcal{L}_{\text{Pang}}$ is mainly picked to act as a references. The loss functions \mathcal{L}_U and \mathcal{L}_O are intended to address

the needs of the two identified passenger types. Finally, MAE^* is intended to represent the one-minute acceptable deviation and give feedback based on that idea. Furthermore, the loss function MAE is very similar to \mathcal{L}_{Pang} and is therefore not included. The same applies to MSE^* which basically is a combination of MAE^* and MSE . The only difference is that MAE^* gives an absolute error outside of the one minute accepted region instead of a squared error, as MSE^* does.

Loss functions
MSE
Pang et al. [3]
U
MAE^*
O

Table 4.2: Loss functions evaluated in this thesis.

The dimensions of the weight matrices associated with the input vector was defined by $\mathbf{U} \in \mathbb{R}^{13 \times 64}$, indicating that the input vector at each time step consists of 13 features and 64 units. The 13 features are further described in section 4.4. Furthermore, the weight matrices associated with the hidden state was assigned the dimension $\mathbf{W} \in \mathbb{R}^{64 \times 64}$ while the last matrix simply holds the $\mathbf{V} \in \mathbb{R}^{64 \times 1}$ dimension. All of the weight matrices were initialized with the method as described by Glorot and Benigo [42] and defined in Equation 2.38. The biases were initialized to zero. Note that the biases as well as the weights are shared across all time steps. The activation functions were the same as in Figure 2.5.

It was empirically found while training that the models tended to overfit. To overcome this issue and prevent the models from overfitting, recurrent dropout of 30% was applied.

Figure 4.3 depicts a graphical representation of the LSTM network architecture. The figure shows how a feature vector is fed to the network at each time step. The attentive reader will question why the sequences does not include the very last stop of the route. The last time step in Figure 4.3 represent the input vector associated with stop $S - 2$, and the output o^{S-2} represents the prediction for the next stop, $S - 1$. It is indeed a good question and there are mainly two reasons why. The first one is that there simply is a considerable amount of missing data on only the last stop for many trips, which greatly reduces the number of available sequences to train and test on. This has probably to do with how the API is configured. Some missed values are expected during a trip since the data is sent with UDP. The problem here is that information on the very last stop often is missing. However, missing data becomes a problem because of the fixed sequence length. The approach taken in this thesis requires the sequences to be of equal length τ without missing information.

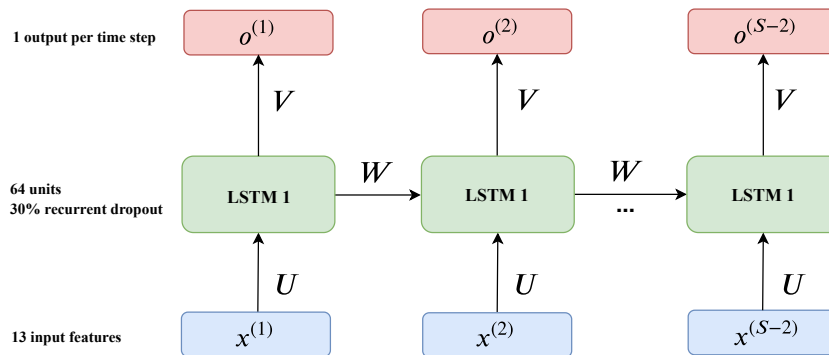


Figure 4.3: Network architecture of the examined models.

4.4 Training

For all routes, a training set were produced with input and target sequences $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_{S-2}\}$ and $\mathbf{Y} = \{\mathbf{a}_2, \dots, \mathbf{a}_{S-1}\}$, respectively. The training procedure is outlined with pseudo code in Algorithm 1. Note that the mini-batch setting causes the model update to occur after the whole batch has been considered. As a point of reference, Masters and Luschi [60] proposed a mini-batch between $batch_size = 2$ and $batch_size = 32$. This is in line with what Benigo [61] suggested in an earlier study by stating that a batch size of 32 is a good default value. Therefore, the batch size were set to 32 when the models in this thesis were trained.

Line nine to twelve in Algorithm 1 goes through each of the sequence pairs and produces an output which is compared to the target value. The loss is accumulated until line 13 is reached. At this point, the gradient is calculated with respect to all weights in the LSTM model. The result is fed to the Adam [25] optimizer which updates the weights of the model.

Algorithm 1 Training procedure

```

1:  $\tau \leftarrow seq\_length$ 
2:  $\eta \leftarrow learning\_rate$ 
3: Training set:  $\chi \leftarrow \{(\mathbf{X}_i, \mathbf{Y}_i), \dots, (\mathbf{X}_i, \mathbf{Y}_i)\}$ 
4: Recurrent state:  $\mathbf{h}_0 \leftarrow initialized(\mathbf{h}_0)$ 
5: repeat
6:    $batch \leftarrow batch\_size$  samples from  $\chi$ 
7:   Loss:  $l \leftarrow 0$ 
8:   LSTM network  $LSTM \leftarrow \mathbf{h}_0$ 
9:   for sequences in batch do
10:    for  $t = 1$  to  $\tau$  do
11:       $\hat{y}_t \leftarrow LSTM(x_t)$ 
12:       $l \leftarrow l + \frac{\mathcal{L}(\mathbf{y}_t, \hat{y}_t)}{\tau}$ 
13:     $\forall w \in LSTM : \frac{\partial l}{\partial w} \leftarrow BPTT(l, w, \tau)$ 
14:     $\forall w \in LSTM : Adam(w, \frac{\partial l}{\partial w}, \eta)$ 
15:     $\nabla \mathbf{h}_0 l \leftarrow BPTT(l, \mathbf{h}_0, \tau)$ 
16:     $Adam(\mathbf{h}_0, \nabla \mathbf{h}_0 l, \eta)$ 
17: until convergence

```

The training algorithm is executed until convergence according to Algorithm 1, but can be halted earlier because of the use of early stopping. This is achieved by allowing many epochs and utilizing early stopping to gain the weights resulting in the lowest validation score. However, it turned out empirically that the parameters for the early stopping as well as the learning rate η needed to be adjusted depending on the model to reach convergence in a good way. A good way here refers to a controlled decline in training and test error without going unnecessarily slow or oscillating between each epoch. For an overview of the static training variables and their values, see Table 4.3.

Parameter	Value
Batch size	32
Optimizer	Adam [25]
Epochs	20 000 (Early stopping)

Table 4.3: Static training parameters and their values.

The training parameters adapted to the characteristics of the route are shown in Table 4.4, including the patience (p) parameter used in early stopping. The values were empirically

found during the training phase by manual tuning. All training parameters were based on the training data.

Rotue	Parameter	
	η	p
70	0.001	150
303	0.001	150
616	0.0001	200
20	0.01	150
119	0.001	150
3	0.0001	200
450	0.001	150
45	0.001	150

Table 4.4: Training parameters grouped by route.

4.5 Feature vector

The input feature vector \mathbf{x}_s is constructed for each stop $s = 1, \dots, S - 2$ and consists of a total of 13 features. The features can be arranged in three categories, spatial, temporal and static.

In the category spatial, there are information included about the position of the current stop and the stop ahead. Let the position of the current stop be denoted as \mathbf{p}_s and the position of the stop ahead as \mathbf{p}_{s+1} such that

$$\mathbf{p}_s = [X_s, Y_s, Z_s]. \quad (4.7)$$

The positions were converted from latitude and longitude coordinates to a Cartesian coordinate system with help of the equation

$$X_s = R_{\oplus} * \cos(lat_s) * \cos(lon_s) \quad (4.8a)$$

$$Y_s = R_{\oplus} * \cos(lat_s) * \sin(lon_s) \quad (4.8b)$$

$$Z_s = R_{\oplus} * \sin(lat_s) \quad (4.8c)$$

where lat_s and lon_s is the latitude and longitude position of stop s and $R_{\oplus} = 6378100 \text{ m}$, which is the nominal earth equator radius. Furthermore, the route distance in kilometers from the current stop to the next was also included in the feature vector as \tilde{d} . The distance information was extracted from the GTFS feed.

For temporal measurements, the departure time from the very first stop d_1 is included as well as the arrival time at the current stop. The reason for only including these is that there under no circumstances can be information about future observations included since the model would easily find and exploit this pattern. Moreover, the model would be useless because it would require giving the model information about future events which is not present during the prediction phase. For instance, the departure time at each stop could not be included since that would give the model invaluable information about future events which is unknown until the event occurs.

Finally, the static variables included are the direction dir of the trip, whether the stop s is a timed stop ts or not, the number of routes \tilde{r} that operates the bus stop, as well as the timetable time for stop s in seconds from midnight. These variable introduces notions of time of the

day, direction and facts information about the bus stop. For instance, \tilde{r} becomes a popularity measure on the stop, suggesting that stops serving many routes may be more busy than less popular stops. The input vector \mathbf{x}_s can be described by

$$\mathbf{x}_s = [d_1, a_s, \mathbf{p}_s, \mathbf{p}_{s+1}, \tilde{d}_{s+1}, dir_s, ts_s, \tilde{r}_s, timetable_s] \quad (4.9)$$

4.6 Predictions

A crucial part is that the models must be able to make predictions in a way that is useful in reality. Recall that it is of interest to predict all remaining stops at every stop and this is the way the CPS works. When a bus has been registered at a bus stop, a new set of predictions is created based on the arrival time. The predictions made by the LSTM models works similarly. The prediction at time step t is encoded as the arrival time in the feature vector for time step $t + 1$. This is repeated recursively until the last input time step τ .

Some of the variables have numerically large values with different scales while some are binary. The binary variables are dir_s and ts_s and they are fed to the network as zero or one. The rest of the features were normalized with Equation 4.10 which simply subtract each variable i with the mean and divide by the standard deviation for that particular variable using the entire dataset.

$$\mathbf{x}_{s,i} = \frac{\mathbf{x}_{s,i} - \mu_i}{\sigma_i} \quad (4.10)$$

This approach is similar to Pang et al. [3] where they also found empirically that there was no real difference compared to a max-mean normalization. However, the normalization causes a problem during the iterative prediction cycle. Recall that the target time and the output is the arrival time in minutes. This was solved by normalizing the predicted arrival time before it is used as input in the next time step. The mean and standard deviation used in the normalization step is calculated from all arrival times in the whole dataset pertaining to the particular route being predicted. The prediction algorithm is described in further detail in Algorithm 2.

Algorithm 2 Prediction algorithm

```

1: Load LSTM model:  $LSTM \leftarrow weights$  and  $biases$ 
2: Test set:  $\chi \leftarrow \{(\mathbf{X}_i, \mathbf{Y}_i), \dots, (\mathbf{X}_i, \mathbf{Y}_i)\}$ 
3: for all trips in  $\chi$  do
4:    $predictions$  at stop  $\leftarrow 0$ 
5:   for  $s$  in  $S - 2$  do
6:     if  $s == S - 2$  then
7:       Predict last:  $\hat{y}_{S-1} \leftarrow LSTM(\mathbf{x}_{S-2})$ 
8:        $prediction_{S-2} \leftarrow \hat{y}_{S-1}$ 
9:     else
10:      Predict rest of the trip
11:      repeat
12:         $\hat{y}_{s+1} \leftarrow LSTM(\mathbf{x}_s)$ 
13:         $\hat{y}_{s+1} \leftarrow \frac{(\hat{y}_{s+1} - \mu)}{\sigma}$ 
14:         $\mathbf{x}_{s+1}[a_{s+1}] \leftarrow \hat{y}_{s+1}$ 
15:         $prediction_s \leftarrow \hat{y}_{s+1}$ 
16:      until last stop

```

This method will thus predict the rest of the stops for the trip at each stop. The CPS and TBP predictions are organized in the same way. The output from CPS is simply collected from

the data and organized per stop. The TBP is constructed by always predicting the scheduled arrival time at each given stop.

4.7 Frameworks and hardware

All models built in this thesis were developed in Keras 2.2.4 [62] which is a high-level neural network library written in Python built on top of Tensorflow [63]. The actual training were performed on an Amazon EC2 instance with the following hardware specification:

- An NVIDIA Tesla K80 Accelerator running a NVIDIA GK210 GPU 12 GiB
- AWS version of Intel's 2.7 GHz Broadwell CPU
- 61 GiB RAM

The weights were stored and transferred from the EC2 instance to a local machine where the predictions were made.



5 Results

This chapter presents the results so that the research questions can be answered. Thus, the models that have been built and trained in this thesis are continuously compared with the performance of the baselines. These comparisons are carried out on the test data which was randomly selected for each individual model. The test data is the same per route in order to later be able to compare the impact of the loss functions and address answers to the research questions. Evaluation metrics have been computed and will be presented for each route.

5.1 Arrival time predictions

The following section will include the results for the set of routes considered. Each route is introduced by a presentation of how well the observations predict the schedule. The figures and tables are associated with an explanation of what is factually shown.

5.1.1 Route 70

Figure 5.1 and Figure 5.2 illustrates the MSE when the timetable is predicted by target values on route 70 for trips in two directions. In each direction, the percentage of observations pertaining to each category is shown. Note that the figures are based on both training data and test data pertaining to route 70 and the respective direction. It provides an opportunity to understand how well the observed target times align with the schedule and thus, illustrates where and how large the potential areas of improvements are. For instance, the bus is either in time or late in a predominant majority of the times the route is driven, in both directions. It is also clear that stop number seven holds the greatest average error for late arrivals and departures, as depicted in Figure 5.2. Also, the stops between stop four and eight in the same figure has no observation where the bus is at most one minute late.

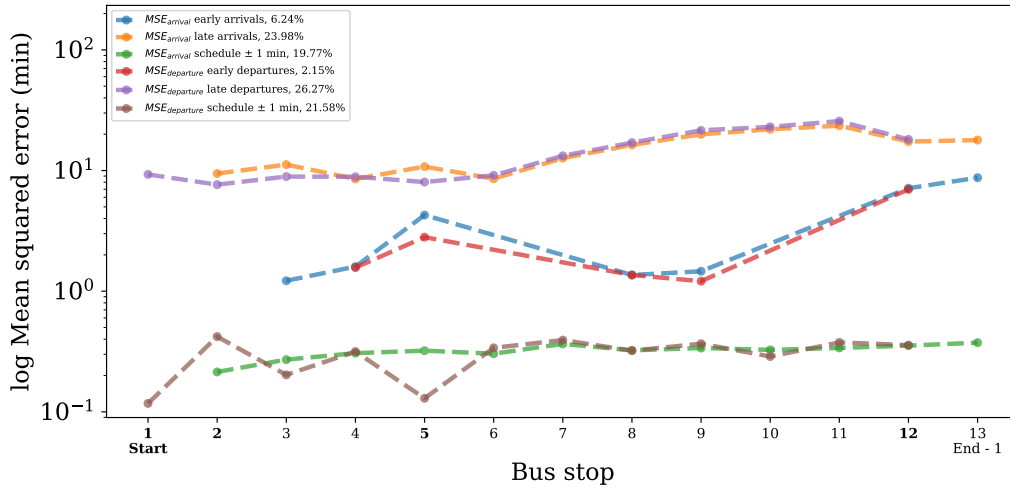


Figure 5.1: MSE for trips from Linköping to Norrköping when the schedule is predicted by observations.

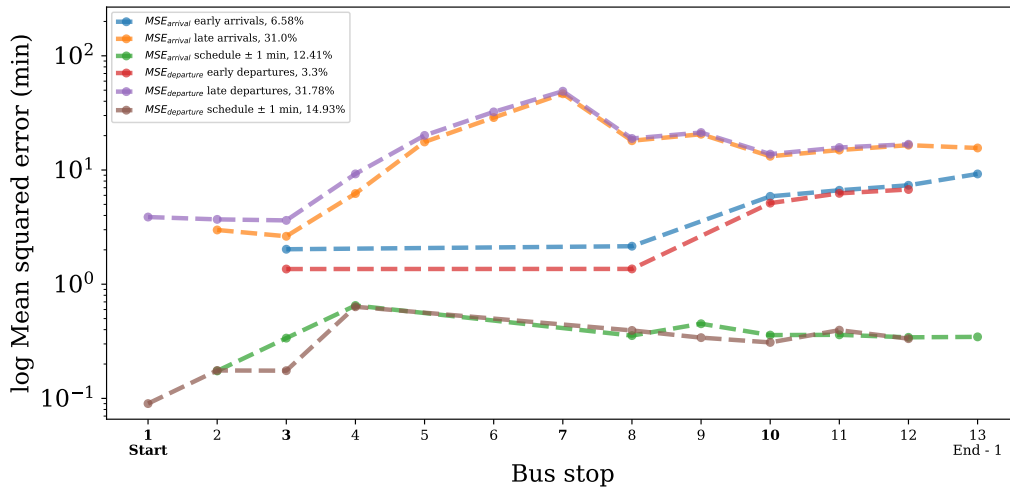


Figure 5.2: MSE for trips from Norrköping to Linköping when the schedule is predicted by observations.

The next illustration, Figure 5.3 depicts a boxplot of the *MAE* prediction error per stop with all three predictions methods included. The CPS, TBP as well as the LSTM model with the loss function MSE. For each stop, a set of predictions are created for the remaining stops. Those predictions are evaluated against the target values and the accumulated error for each method is shown from the stop where the predictions were made. The dashed lines marks the 95th percentile, indicating that 95% of the accumulated prediction errors falls below those lines. Figure 5.3 shows that both CPS and the LSTM model manage to overall reduce the error more than TBP.

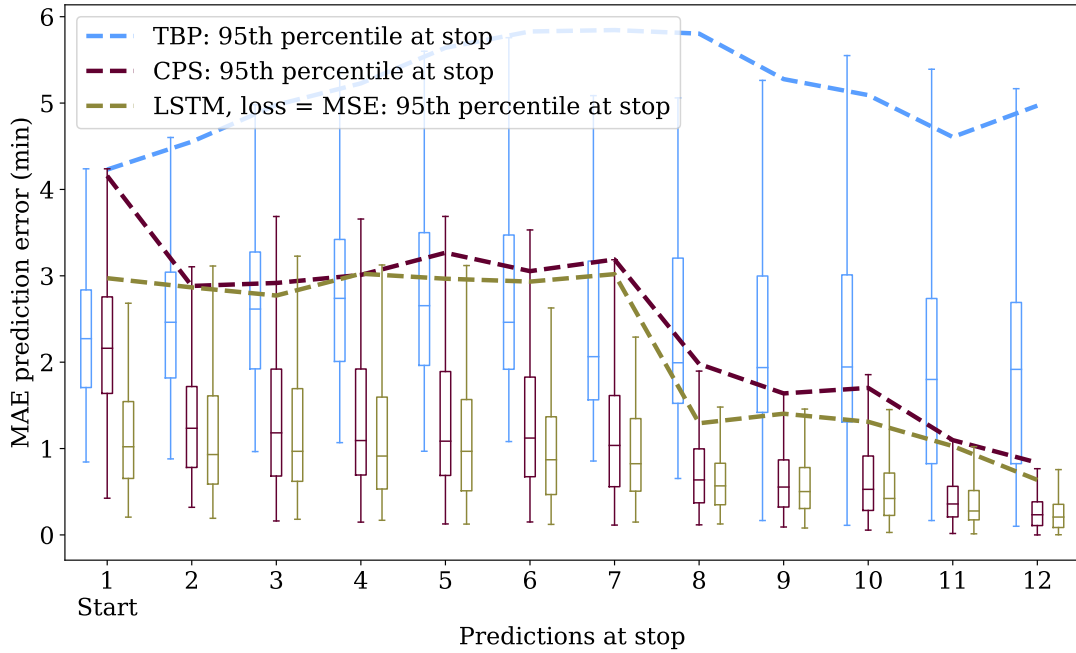


Figure 5.3: Prediction errors on the test set with 159 sequences as measured in *MAE*. The LSTM model is trained with MSE as loss.

A full comparison of all considered metrics is showed in Table 5.1. Note that LSTM improvements compares the best LSTM model with TBP and CPS, respectively. The LSTM model in each metric with the lowest mean error is marked bold. All predictions are divided into three categories. Those are *correct*, *underestimated* and *overestimated*. A prediction is considered to be correct if and only if $|a - \hat{a}| \leq 1$. Furthermore, a prediction falls into the underestimated category if and only if the prediction is underestimated more than one minute. Similarly, the overestimated category contains predictions overestimated more than one minute. The percentage shows how many of the total number of predictions that ends up in each category.

Method	Evaluation metrics					Prediction category (%)			
	Minutes				%	Correct	Under-estimated	Over-estimated	
	<i>RMSE</i>	<i>RMSE*</i>	<i>MAE</i>	<i>MAE*</i>	<i>MAPE</i>				
TBP	2.86 ± 1.47	2.65 ± 1.53	2.55 ± 1.46	1.67 ± 1.40	8.80 ± 3.36	25.4	57.9	16.7	
CPS	1.29 ± 0.74	0.95 ± 0.78	1.14 ± 0.63	0.50 ± 0.57	3.86 ± 1.33	52.4	24.1	23.5	
LSTM	MSE	1.03 ± 0.75	0.69 ± 0.79	0.91 ± 0.63	0.34 ± 0.56	2.83 ± 1.33	61.9	18.0	20.1
	Pang	1.01 ± 0.73	0.66 ± 0.78	0.89 ± 0.62	0.32 ± 0.54	2.74 ± 1.32	63.7	18.4	17.9
	U	1.32 ± 0.91	1.0 ± 0.96	1.16 ± 0.77	0.55 ± 0.69	3.35 ± 1.62	52.7	43.5	3.8
	MAE*	1.17 ± 0.71	0.82 ± 0.76	1.02 ± 0.61	0.44 ± 0.53	3.19 ± 1.47	57.9	13.4	28.7
	O	1.42 ± 0.93	1.37 ± 1.08	1.18 ± 0.77	0.57 ± 0.70	3.45 ± 1.64	43.9	5.4	50.7
LSTM Improvements	TBP	64.7%	75.1%	65.1%	80.8%	68.9%	150.8%	-	-
	CPS	21.7%	30.5%	21.9%	36.0%	29.0%	21.6%	-	-

Table 5.1: Evaluation results for route 70, $N = 159$.

The prediction categories shown in Table 5.1 shows how many of the total predictions that falls into the acceptable range and thus, are considered to be correct as well as the distribution of the errors for each method. This is illustrated further in more detail in Figure 5.4. All the arrival time predictions produced at each stop, starting from the very first stop, have been evaluated against the target values. This makes it possible to follow how the ratio between the three categories develops over time as a complement to the overall outcome.

The type of error at each stop for the whole route by the CPS and TBP is illustrated in Figure 5.4. Furthermore, the type of error with *U* and *O* as loss functions is depicted in

Figure 5.5. The effect of the loss function is clearly visible as the predictions by U is either correct or underestimated in 96.2% of the cases, while the predictions produced by O on the other hand is either correct or overestimated 94.6% of the times.

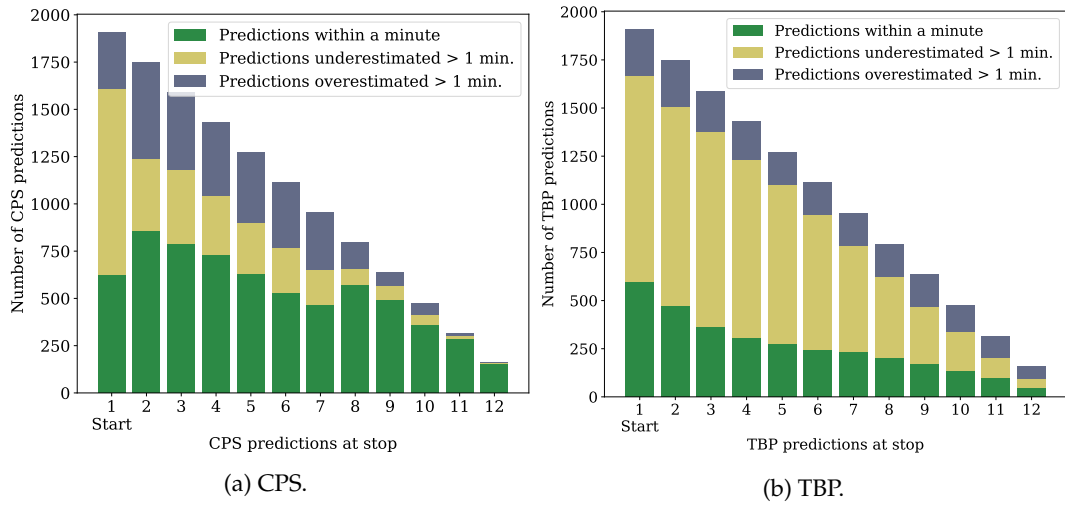


Figure 5.4: Type of error at each stop for the CPS and TBP on route 70.

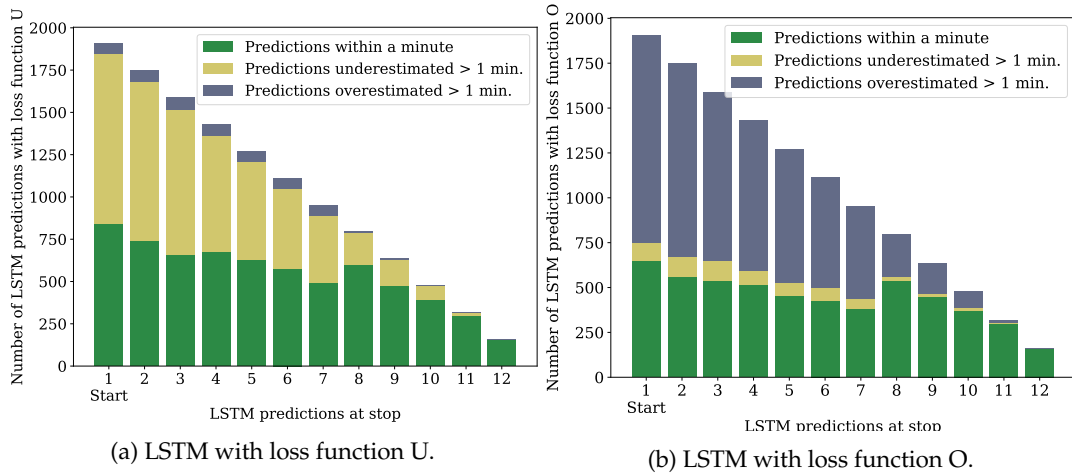


Figure 5.5: Type of error at each stop on route 70.

5.1.2 Route 303

Similarly to route 70, Figure 5.6 and Figure 5.7 depicts the MSE when the timetable is predicted by the target values for both directions. This time, it is the city route with number 303 that is evaluated.

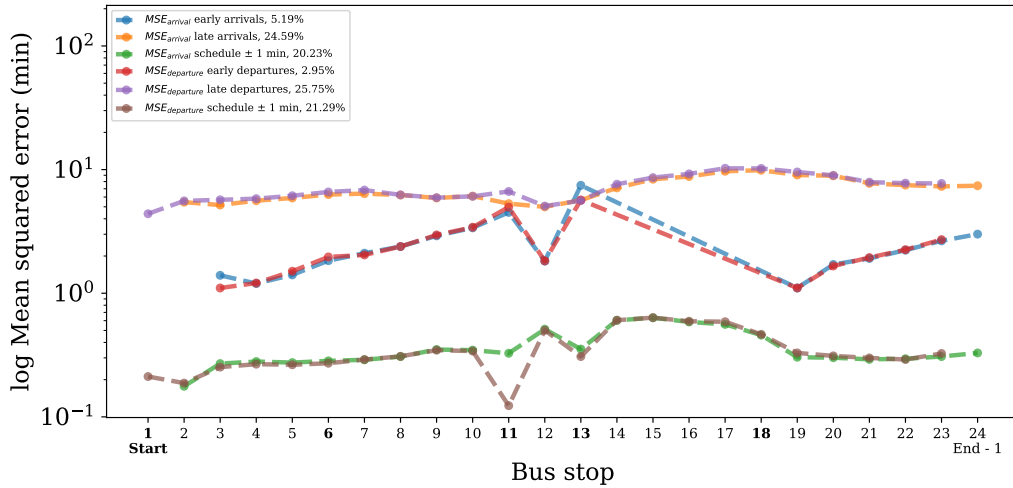


Figure 5.6: MSE for trips on route 303 when the schedule is predicted by target values.

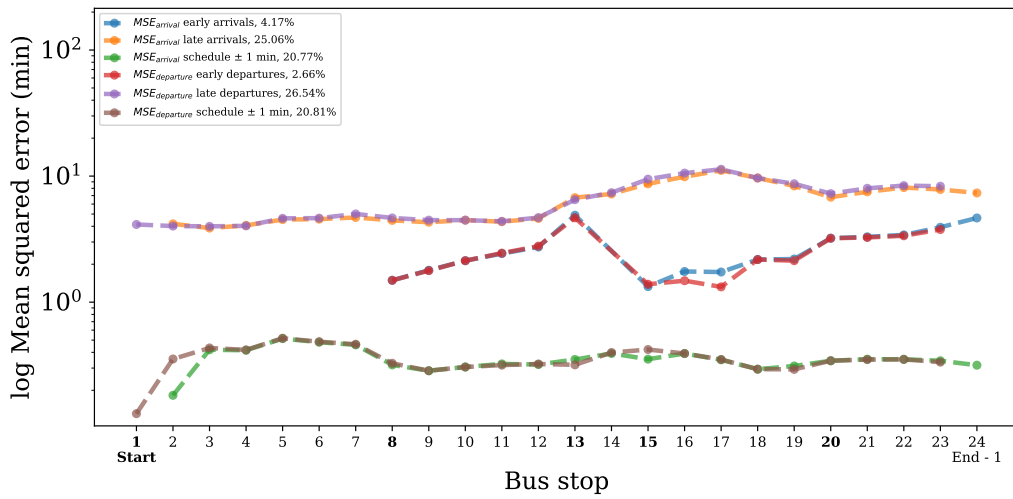


Figure 5.7: MSE for trips on route 303 when the schedule is predicted by target values.

Figure 5.8 depicts how the CPS, TBP and the LSTM model with loss function Pang et al. [3] performs on route 303. Again, the model using that loss function managed to produce the best results in all evaluation metrics as shown in Table 5.2.

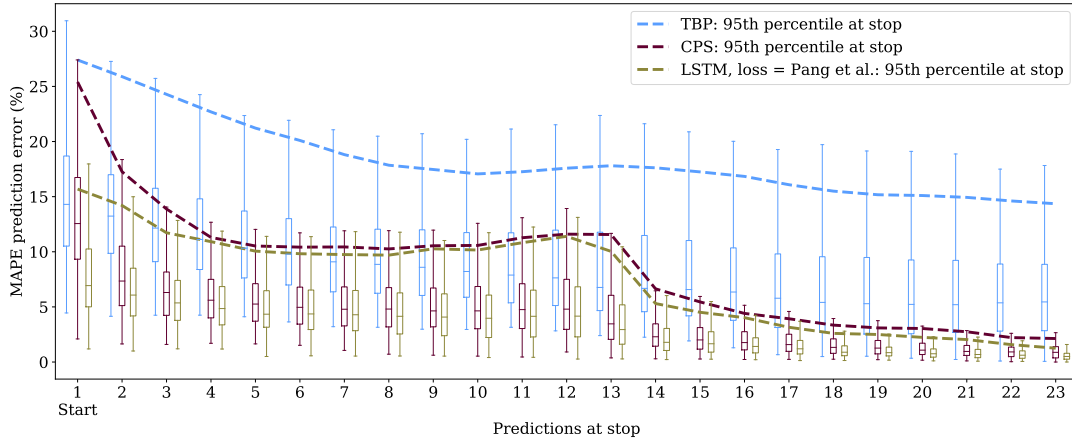


Figure 5.8: Prediction errors on the test set with 409 sequences as measured in *MAPE*.

Method	Evaluation metrics					Prediction category (%)			
	Minutes				%	Correct	Under-estimated	Over-estimated	
	RMSE	RMSE*	MAE	MAE*	MAPE				
TBP	1.83 ± 1.05	1.53 ± 1.14	1.69 ± 1.04	0.87 ± 0.93	9.18 ± 4.26	37.7	50.0	12.4	
CPS	0.85 ± 0.35	0.51 ± 0.37	0.75 ± 0.31	0.21 ± 0.23	4.31 ± 1.55	66.1	16.9	17.0	
LSTM	MSE	0.74 ± 0.36	0.43 ± 0.38	0.64 ± 0.32	0.17 ± 0.22	3.52 ± 1.47	70.9	17.2	11.9
	Pang	0.75 ± 0.36	0.44 ± 0.38	0.64 ± 0.32	0.17 ± 0.23	3.52 ± 1.47	71.3	16.1	12.6
	U	1.02 ± 0.55	0.73 ± 0.58	0.87 ± 0.49	0.35 ± 0.38	4.67 ± 2.09	61.5	37.5	0.9
	MAE*	1.19 ± 0.48	0.86 ± 0.52	1.02 ± 0.41	0.36 ± 0.30	5.77 ± 2.46	54.6	6.4	39.0
	O	0.86 ± 0.39	0.55 ± 0.42	0.74 ± 0.35	0.23 ± 0.26	4.23 ± 2.09	65.4	5.6	29.0
LSTM	TBP	59.6%	71.9%	62.1%	80.5%	61.7%	89.1%	-	-
Improvements	CPS	12.9%	15.7%	14.7%	19.0%	18.3%	7.9%	-	-

Table 5.2: Evaluation results for route 303, $N = 409$.

The distribution of correct predictions and errors for the CPS and TBP is illustrated by Figure 5.9, while Figure 5.10 depicts the distribution for the LSTM models with loss function U and O , respectively. The figures shows that the predictions produced by the CPS on the first stop underestimates a larger proportion than the rest of the predictions, where the errors gets fairly evenly distributed. Also, O predicts more correct than U but also gets a minor portion of unwanted prediction errors.

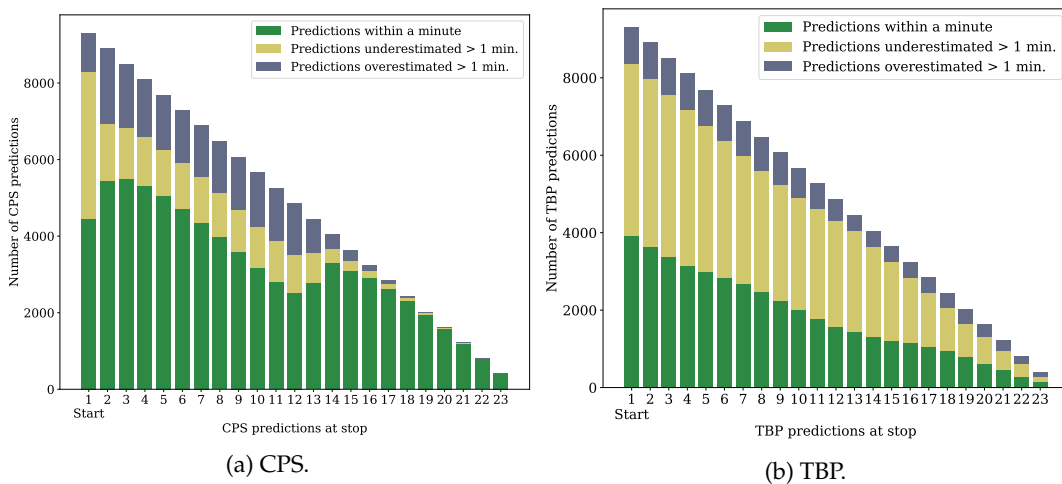


Figure 5.9: Type of error at each stop for the CPS and TBP on route 303.

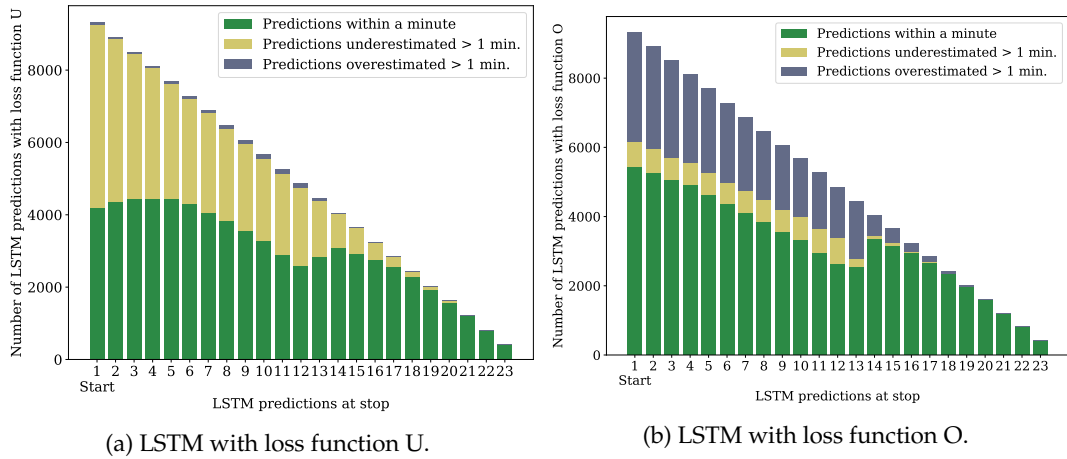


Figure 5.10: Type of error at each stop on route 303.

5.1.3 Route 616

Figure 5.12 and Figure 5.11 depicts the MSE when the timetable is predicted by target values for both directions on route 616 between Linköping and Borensberg. The figures illustrates a similar trend as on the previous routes examined. The bus is either in time, a bit early on a few occasions or quite late compared to the timetable. For all stops visited, the bus late to approximately 74% of the stops in one direction and 63% in the opposite direction.

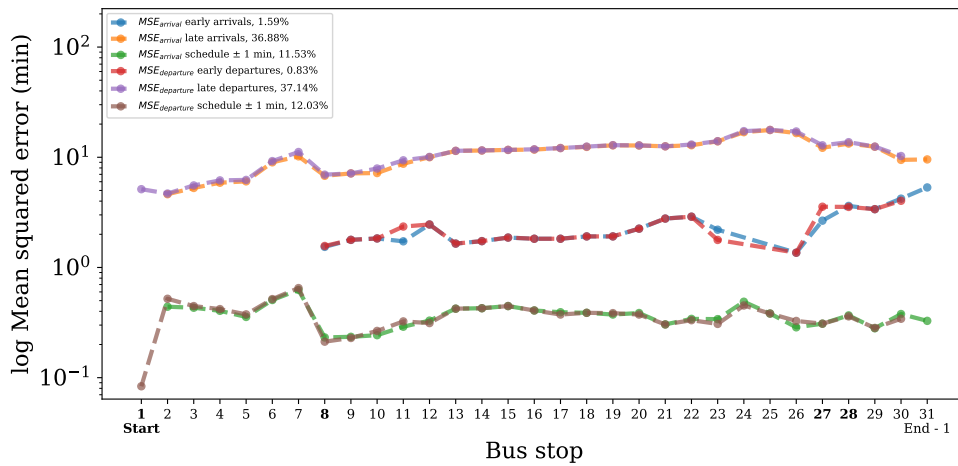


Figure 5.11: MSE on route 616 for trips from Linköping to Borensberg when the schedule is predicted by target values.

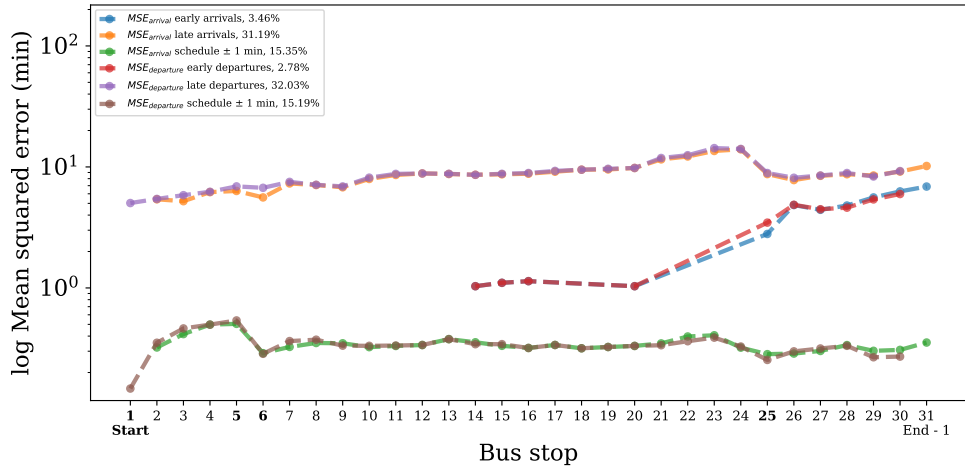


Figure 5.12: MSE on route 616 for trips from Borensberg to Linköping when the schedule is predicted by target values.

Figure 5.13 depicts how the CPS, TBP and the LSTM model with loss function Pang et al. [3] performs on route 616 as measured in MAPE. This time,

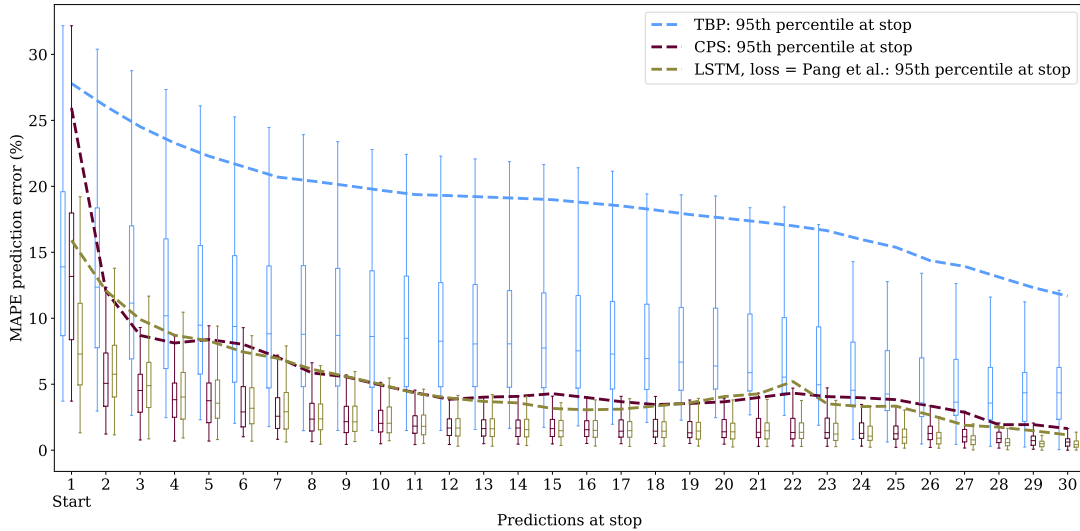


Figure 5.13: Prediction errors on the test set with 108 trips as measured in MAPE.

Table 5.3 shows the evaluation metrics.

Method	Evaluation metrics					Prediction category (%)			
	Minutes				%	Correct	Under-estimated	Over-estimated	
	RMSE	RMSE*	MAE	MAE*	MAPE				
TBP	2.49 ± 1.47	2.26 ± 1.55	2.32 ± 1.48	1.45 ± 1.4	8.55 ± 4.49	26.6	65.3	8.1	
CPS	0.79 ± 0.42	0.42 ± 0.45	0.69 ± 0.32	0.17 ± 0.25	2.65 ± 0.92	73.0	13.6	13.4	
LSTM	MSE	0.75 ± 0.48	0.40 ± 0.51	0.64 ± 0.37	0.16 ± 0.28	2.32 ± 1.01	73.7	15.9	10.4
	Pang	0.75 ± 0.43	0.40 ± 0.46	0.64 ± 0.33	0.15 ± 0.26	2.41 ± 0.96	72.7	8.8	18.5
	U	1.02 ± 0.62	0.73 ± 0.66	0.86 ± 0.46	0.34 ± 0.38	3.04 ± 1.32	62.8	36.4	0.8
	MAE*	1.49 ± 0.54	1.20 ± 0.59	1.30 ± 0.45	0.59 ± 0.38	4.73 ± 1.67	43.2	34.5	22.3
	O	0.90 ± 0.48	0.57 ± 0.52	0.77 ± 0.37	0.23 ± 0.29	2.85 ± 1.21	66.6	5.6	27.8
LSTM Improvements	TBP	69.9%	82.3%	72.4%	89.7%	72.9%	177.1%	-	-
	CPS	5.0%	4.8%	7.2%	11.8%	12.5%	1.0%	-	-

Table 5.3: Evaluation results for route 616, N = 108.

The distribution prediction categories per stop for CPS and the LSTM model with loss function Pang et al. [3] is illustrated by Figure 5.14. The predictions produced by CPS on the very first stop is tilted towards underestimations while the rest of the prediction errors are evenly distributed, compared to the LSTM model which is overestimating more than underestimating in the first set of predictions.

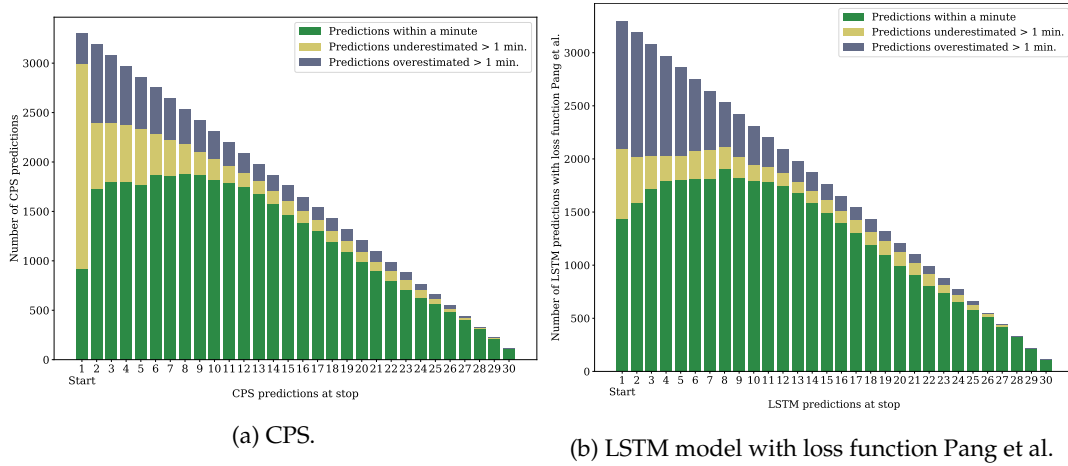


Figure 5.14: Type of error at each stop for the CPS and the LSTM model loss function with Pang et al. on route 616.

5.1.4 Route 20

Route 20 is a relatively short city route in Linköping with few stops. Figure 5.15 and Figure 5.16 depicts the MSE when the timetable is predicted by the target values for both directions.

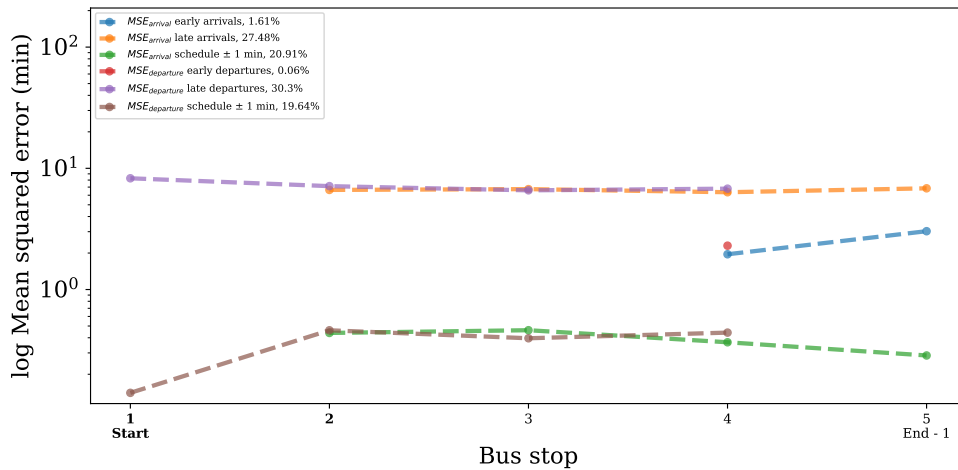


Figure 5.15: MSE on route 20, located in Linköping when the schedule is predicted by target values.

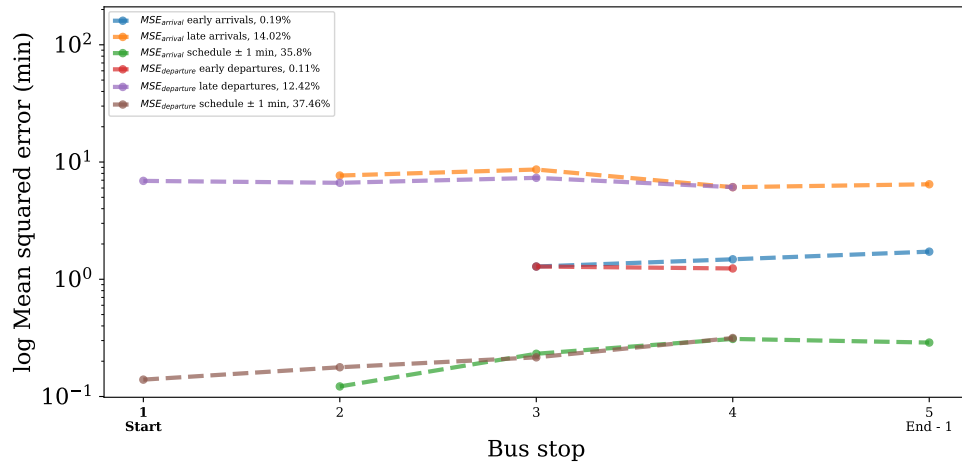


Figure 5.16: MSE on route 20, located in Linköping when the schedule is predicted by target values.

Figure 5.17 depicts how the CPS, TBP and the LSTM model with loss function Pang et al. [3] performs on route 20.

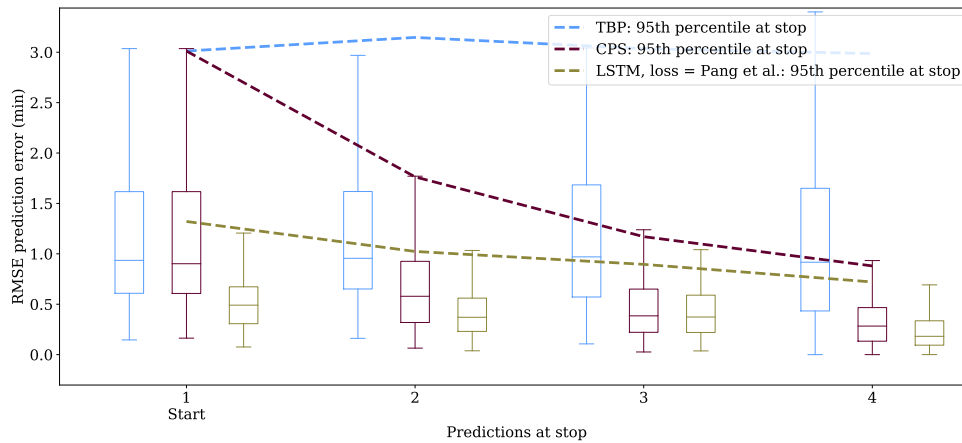


Figure 5.17: Prediction errors on the test set with 145 trips as measured in RMSE.

Table 5.4 shows the evaluation metrics.

Method	Evaluation metrics					Prediction category (%)			
	Minutes				%	Correct	Under-estimated	Over-estimated	
	RMSE	RMSE*	MAE	MAE*	MAPE				
TBP	1.28 ± 1.12	0.86 ± 1.23	1.22 ± 1.12	0.51 ± 0.98	16.25 ± 10.34	57.4	40.2	2.4	
CPS	0.71 ± 0.41	0.33 ± 0.42	0.66 ± 0.40	0.17 ± 0.30	10.18 ± 5.85	75.6	17.4	7.0	
LSTM	MSE	0.44 ± 0.26	0.09 ± 0.22	0.40 ± 0.23	0.03 ± 0.11	6.53 ± 4.33	91.3	4.8	3.9
	Pang	0.42 ± 0.23	0.08 ± 0.20	0.39 ± 0.21	0.03 ± 0.09	6.33 ± 4.29	92.6	4.8	2.6
	U	0.49 ± 0.31	0.15 ± 0.28	0.45 ± 0.27	0.06 ± 0.14	7.07 ± 4.86	85.7	13.6	0.8
	MAE*	0.60 ± 0.26	0.18 ± 0.21	0.56 ± 0.24	0.06 ± 0.09	10.74 ± 7.87	81.0	2.7	16.3
O	0.47 ± 0.25	0.12 ± 0.19	0.43 ± 0.23	0.04 ± 0.08	7.50 ± 6.05	86.6	2.5	10.9	
LSTM Improvements	TBP	67.2%	90.7%	68.0%	94.1%	61.0%	61.3%	-	-
CPS	40.8%	75.8%	40.9%	82.4%	37.8%	22.5%	-	-	

Table 5.4: Evaluation results for route 20, $N = 145$.

The distribution prediction categories per stop for CPS and the LSTM model with loss function Pang et al. [3] is illustrated by Figure 5.18. The amount of correct predictions is increased by the LSTM model for all stops.

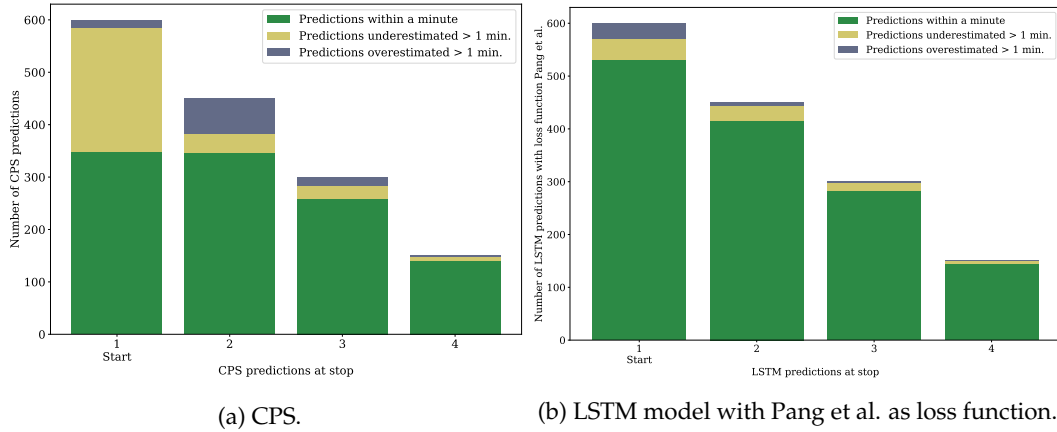


Figure 5.18: Type of error at each stop for the CPS and the LSTM model with loss function Pang et al. on route 20.

5.1.5 Route 119

Route 119 is a city route in Norrköping. Figure 5.19 illustrates that 51.93% of the arrivals and departures are within the one minute acceptable range but only 20.94% in the other direction, as depicted by Figure 5.20. The figures also show that the trend of increased errors towards the end of the route for early and late arrivals and departures continues on this route as well, indicating that the relevance of the timetable for those instances decreases as the target values diverge more and more.

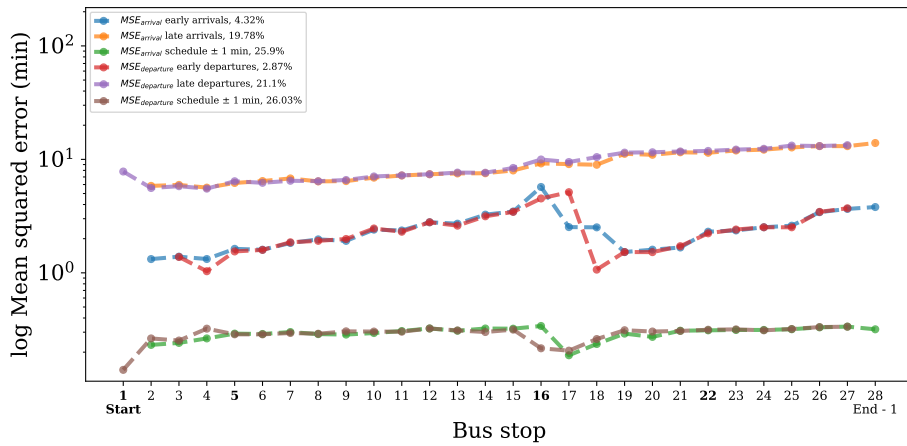


Figure 5.19: MSE on route 119, located in Norrköping when the schedule is predicted by target values.

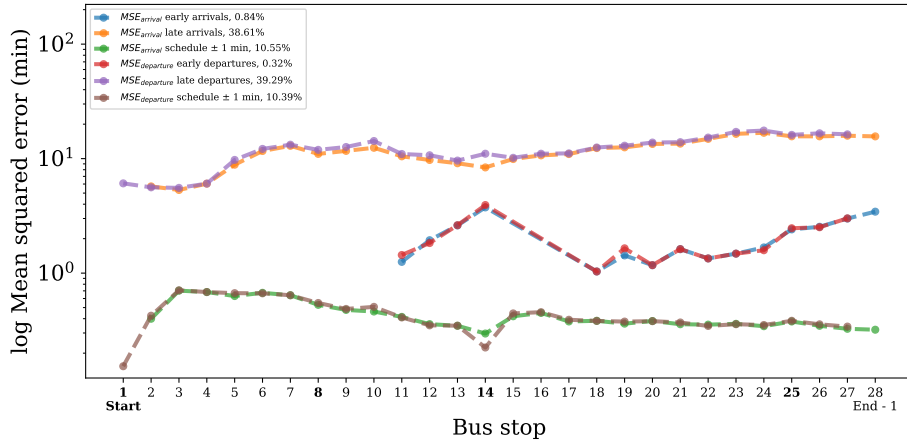


Figure 5.20: MSE on route 119 for trips in Linköping when the schedule is predicted by target values.

Now when the conditions on the route is known, Figure 5.21 depicts how the CPS, TBP and the LSTM model with loss function *MSE* performs on route 119. This time, the evaluation metric in the figure is *RMSE*. The summary of all evaluation metrics in Table 5.5 shows that the LSTM models with loss functions *MSE* and Pang et al. [3] are quite even. They are also able to put more prediction in the correct category than both TBP and CPS. One of the reasons why is illustrated Figure 5.22, where CPS heavily underestimates the arrival at the first stop compared to the LSTM model.

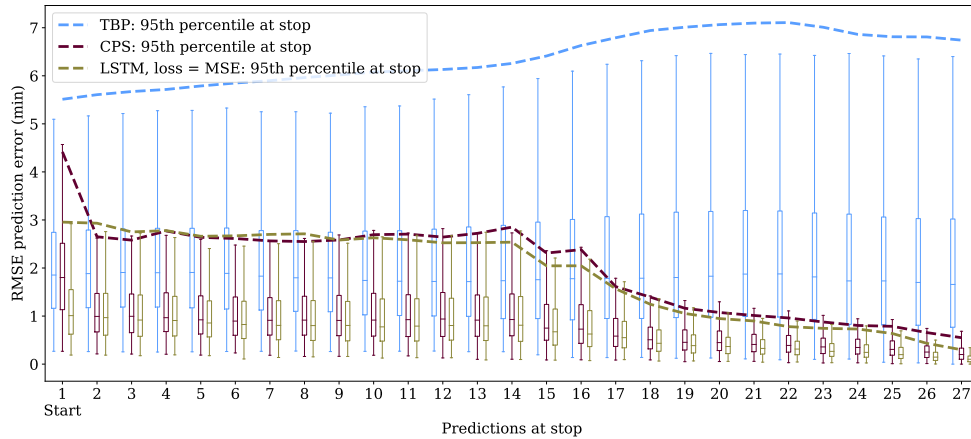


Figure 5.21: Prediction errors on the test set with 716 trips as measured in *RMSE*.

Method	Evaluation metrics					Prediction category (%)			
	Minutes				%	Correct	Under-estimated	Over-estimated	
	<i>RMSE</i>	<i>RMSE*</i>	<i>MAE</i>	<i>MAE*</i>	<i>MAPE</i>				
TBP	2.32 ± 1.86	2.05 ± 1.95	2.21 ± 1.85	1.38 ± 1.74	8.87 ± 5.89	33.3	62.0	4.7	
CPS	0.88 ± 0.43	0.53 ± 0.47	0.77 ± 0.38	0.24 ± 0.30	3.31 ± 1.23	66.9	17.8	15.3	
LSTM	<i>MSE</i>	0.78 ± 0.47	0.45 ± 0.51	0.67 ± 0.4	0.20 ± 0.31	2.71 ± 1.30	72.1	15.3	12.6
	Pang	0.78 ± 0.47	0.46 ± 0.50	0.68 ± 0.4	0.20 ± 0.31	2.75 ± 1.34	71.4	13.4	15.2
	U	1.32 ± 0.7	1.03 ± 0.75	1.13 ± 0.6	0.54 ± 0.52	4.37 ± 1.91	51.9	47.6	0.5
	<i>MAE*</i>	3.98 ± 0.87	3.81 ± 0.89	3.44 ± 0.75	2.55 ± 0.72	13.9 ± 3.54	15.1	6.8	78.2
	O	1.07 ± 0.5	0.76 ± 0.54	0.93 ± 0.43	0.37 ± 0.35	3.81 ± 1.69	57.0	4.8	38.2
LSTM Improvements	TBP	66.4%	78.0%	69.7%	85.5%	69.4%	116.5%	-	-
	CPS	11.4%	15.1%	13.0%	16.7%	18.1%	7.8%	-	-

Table 5.5: Evaluation results for route 119, $N = 716$.

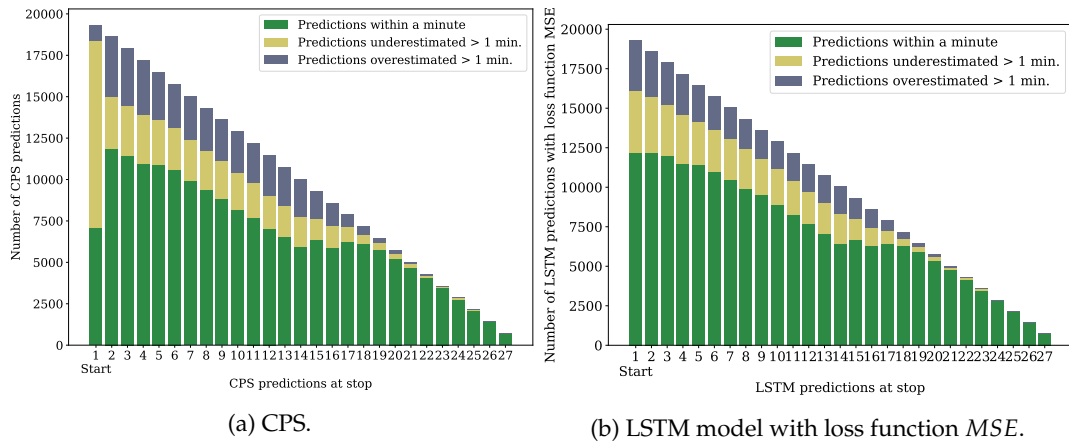


Figure 5.22: Type of error at each stop for the CPS and the LSTM model with loss function *MSE* on route 119.

5.1.6 Route 3

Route 3 is a city route in Linköping. Figure 5.23 and Figure 5.24 depicts the *MSE* when the timetable is predicted by the observations for both directions. With this in mind, Figure 5.25 depicts how the CPS, TBP and the LSTM model with loss function *MSE* performs on route 3. The *RMSE* prediction error for CPS at the first stop is very similar to TBP. For the rest of the stops, both CPS and the LSTM model with loss function *MSE* manage to reduce the error over time, where the LSTM model is reducing the error the most. This is also shown in Table 5.6, which contains all evaluation metrics for route 3.

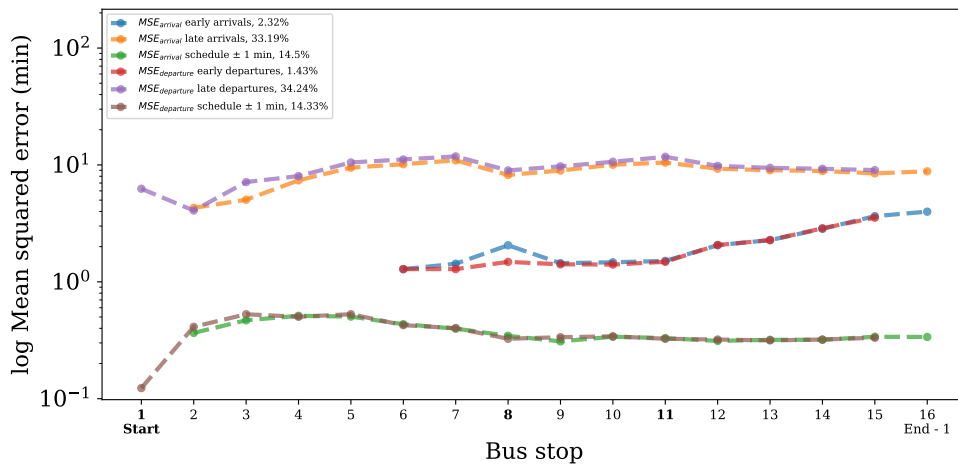


Figure 5.23: *MSE* on route 3, located in Linköping, when the schedule is predicted by target values.

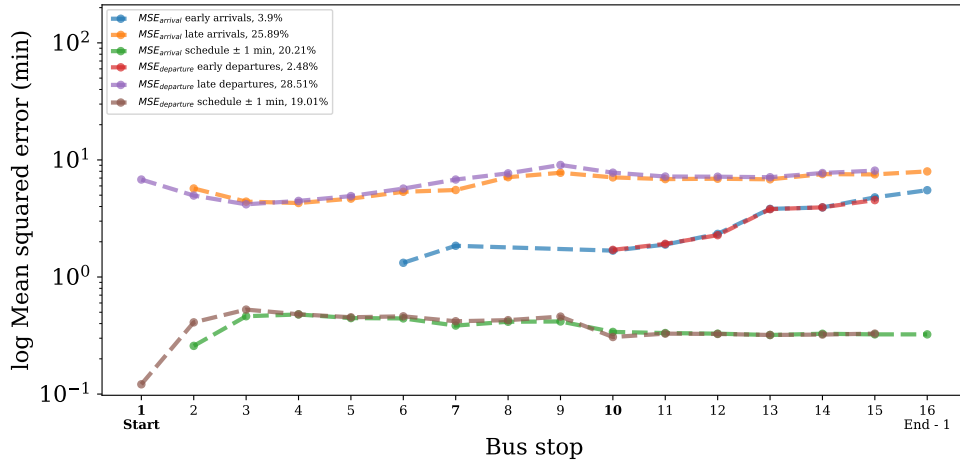


Figure 5.24: MSE on route 3, located in Linköping, when the schedule is predicted by target values.

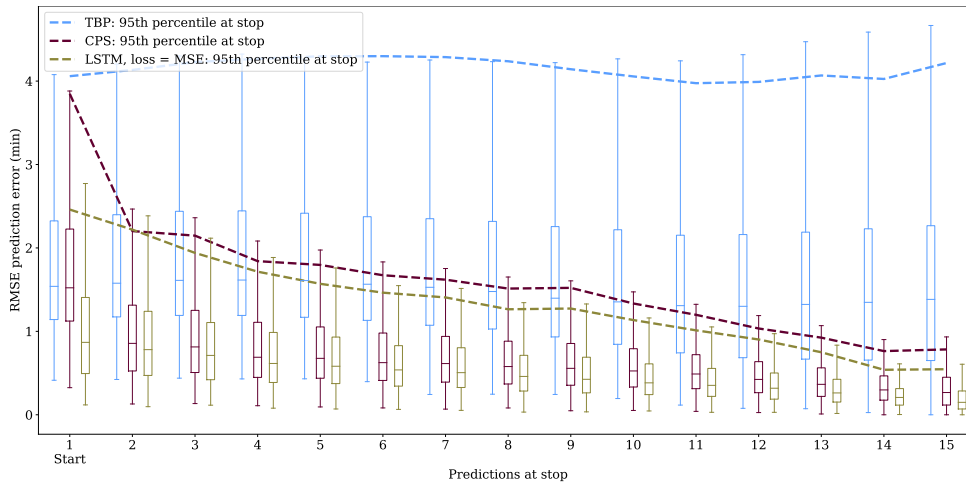


Figure 5.25: Prediction errors on the test set with 1478 trips as measured in RMSE.

Method	Evaluation metrics					Prediction category (%)			
	Minutes				%	Correct	Under-estimated	Over-estimated	
	RMSE	RMSE*	MAE	MAE*	MAPE				
TBP	1.82 ± 1.24	1.5 ± 1.33	1.72 ± 1.26	0.90 ± 1.15	12.56 ± 6.23	35.1	54.6	10.3	
CPS	0.74 ± 0.33	0.36 ± 0.35	0.66 ± 0.30	0.15 ± 0.2	5.17 ± 2.05	72.9	15.7	11.3	
LSTM	MSE	0.58 ± 0.30	0.24 ± 0.30	0.51 ± 0.26	0.09 ± 0.17	3.82 ± 1.94	80.5	7.4	12.1
	Pang	0.56 ± 0.29	0.22 ± 0.29	0.49 ± 0.25	0.09 ± 0.16	3.74 ± 1.84	81.0	6.4	12.6
	U	0.89 ± 0.46	0.57 ± 0.49	0.77 ± 0.39	0.26 ± 0.28	5.28 ± 2.30	64.9	34.6	0.5
	MAE*	1.19 ± 0.49	0.85 ± 0.53	1.06 ± 0.43	0.42 ± 0.32	8.42 ± 4.33	50.0	10.3	39.7
	O	0.69 ± 0.35	0.35 ± 0.36	0.6 ± 0.3	0.14 ± 0.19	4.47 ± 2.35	74.6	3.1	22.3
LSTM	TBP	69.2%	85.3%	71.5%	90.0%	70.2%	157.1%	-	-
Improvements	CPS	24.3%	38.9%	25.8%	40.0%	27.7%	11.1%	-	-

Table 5.6: Evaluation results for route 3, $N = 1478$.

The distribution prediction categories per stop for CPS and the LSTM model with MSE as loss function is illustrated by Figure 5.26. The figures shows that one of the reasons why the best LSTM model manage to improve the number of correct predictions with 11.1% is the

fact that CPS underestimates a large proportion of the predictions produced at the first stop compared to the LSTM model.

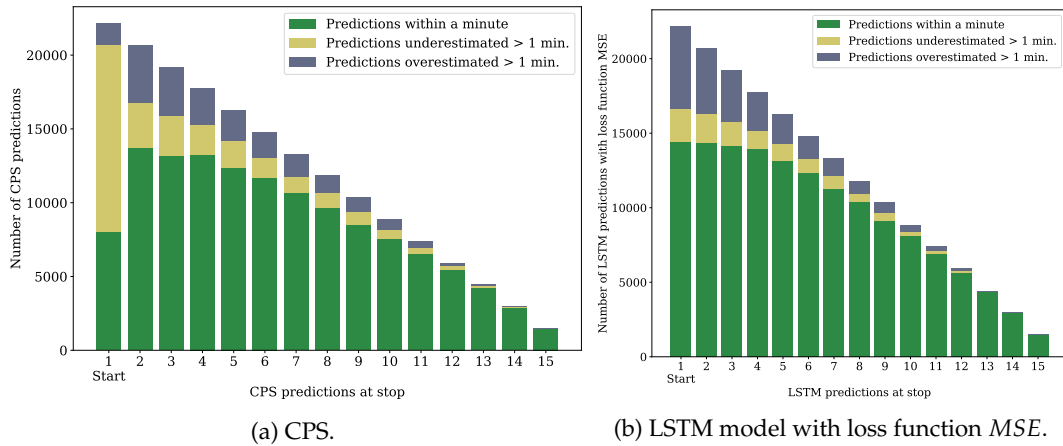


Figure 5.26: Type of error at each stop for the CPS and the LSTM model with MSE as loss function on route 3.

5.1.7 Route 450

Route 450 is a route between the cities Norrköping and Söderköping. Figure 5.27 and Figure 5.28 depicts the MSE when the timetable is predicted by target values for both directions. The figures show that 30.11% if all arrivals and departures are more than one minute earlier than the schedule while the corresponding value for the other direction is 5.82%. Also, only 18.19% of all arrivals and departure are within the acceptable region according to Figure 5.27 compared to the other direction, where 39.1% of all arrivals and departures are most one minute early or late.

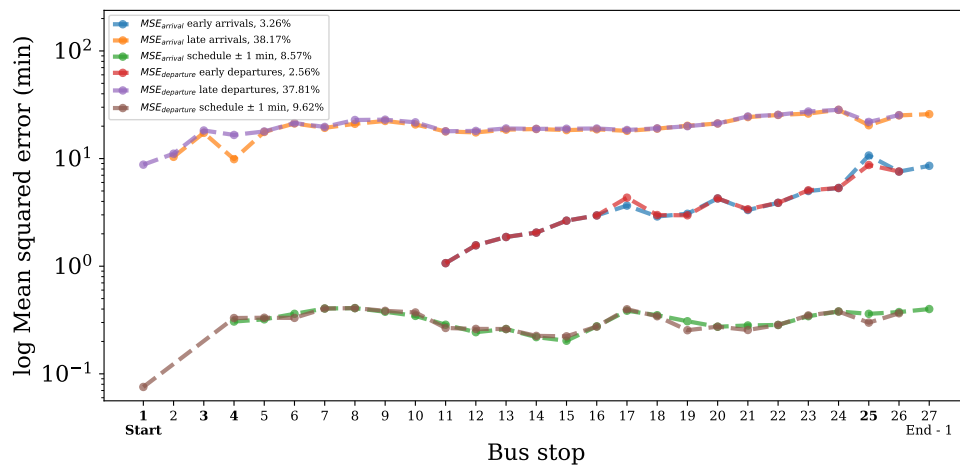


Figure 5.27: MSE on route 450 from Norrköping to Söderköping, when the schedule is predicted by target values.

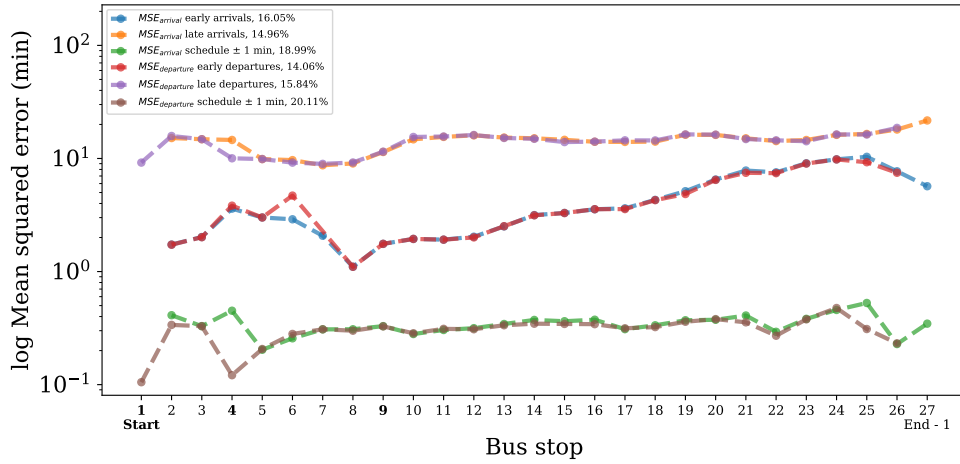


Figure 5.28: MSE on route 450 from Söderköping to Norrköping, when the schedule is predicted by target values.

Figure 5.25 depicts how the CPS, TBP and the LSTM model with Pang et al. [3] as loss function performs on route 450. CPS and TBP are identical on the first stop while the LSTM model is able to reduce the accumulated *MAPE* error.

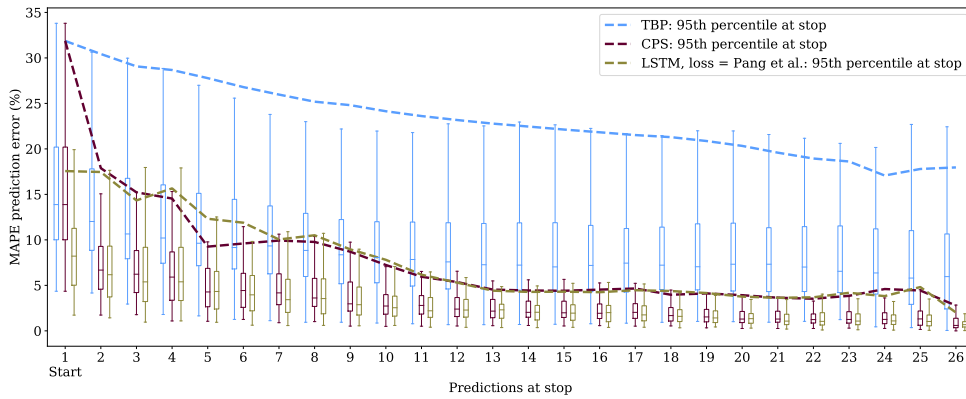


Figure 5.29: Prediction errors on the test set with 104 trips as measured in *MAPE*.

Table 5.7 shows the evaluation metrics. It does not differ much between the best LSTM model and the CPS although there is a slight improvement. Again, one reason for the difference between the best LSTM model and CPS in correct predictions is illustrated in Figure 5.30, which displays that the largest difference is at the first stop.

Method	Evaluation metrics					Prediction category (%)			
	Minutes				%	Correct	Under-estimated	Over-estimated	
	<i>RMSE</i>	<i>RMSE*</i>	<i>MAE</i>	<i>MAE*</i>	<i>MAPE</i>				
TBP	3.05 ± 2.14	2.84 ± 2.22	2.88 ± 2.18	1.99 ± 2.10	9.95 ± 6.02	23.8	52.8	23.3	
CPS	1.14 ± 0.48	0.80 ± 0.53	1.00 ± 0.43	0.39 ± 0.35	3.71 ± 1.27	58.0	22.2	19.8	
LSTM	MSE	1.10 ± 0.54	0.77 ± 0.59	0.95 ± 0.47	0.37 ± 0.40	3.42 ± 1.51	60.2	18.4	21.5
	Pang	1.07 ± 0.59	0.75 ± 0.63	0.92 ± 0.52	0.36 ± 0.43	3.29 ± 1.60	62.0	17.1	20.9
	U	1.70 ± 0.88	1.44 ± 0.93	1.45 ± 0.76	0.81 ± 0.65	4.88 ± 2.14	47.2	51.4	1.4
	MAE*	1.84 ± 0.68	1.57 ± 0.73	1.58 ± 0.58	0.86 ± 0.51	5.69 ± 2.27	39.7	9.4	50.9
	O	1.18 ± 0.51	0.86 ± 0.56	1.02 ± 0.45	0.42 ± 0.38	3.75 ± 1.63	56.4	5.8	37.8
LSTM Improvements	TBP	64.9%	73.6%	68.1%	81.9%	66.9%	169.6%	-	-
CPS	6.1%	6.3%	8.0%	7.7%	11.3%	6.9%	-	-	

Table 5.7: Evaluation results for route 450, $N = 104$.

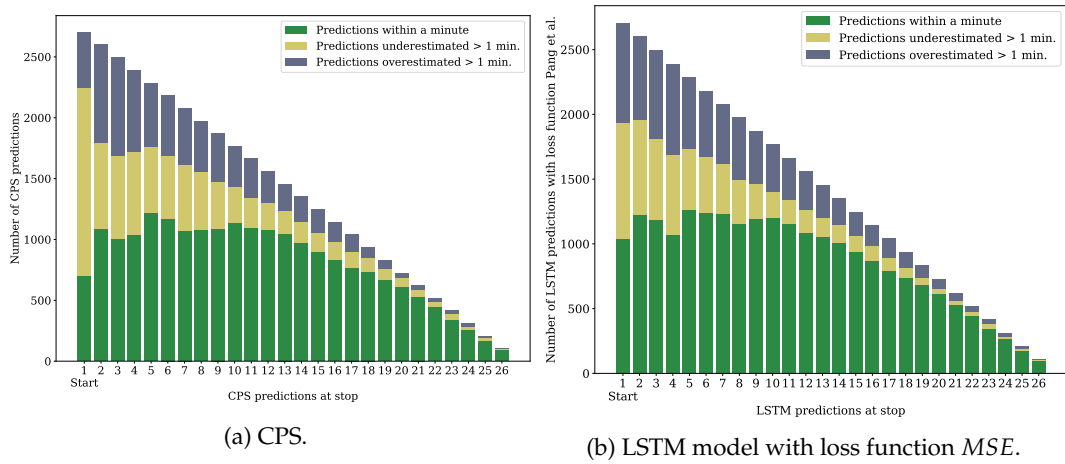


Figure 5.30: Type of error at each stop for the CPS and the LSTM model with loss function MSE on route 450.

5.1.8 Route 45

Finally, the last route included is route 45. This route is also located between the cities Norrköping and Söderköping. One difference between this route and route 450 is that there are fewer stops, indicating that the distance between each stop is longer. Figure 5.31 and Figure 5.32 depicts the MSE when the timetable is predicted by the observations for both directions.

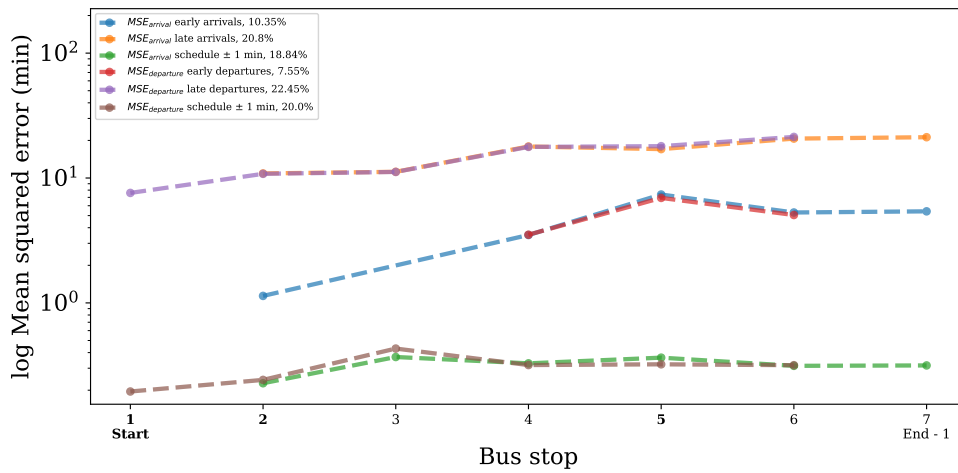


Figure 5.31: MSE on route 45 from Norrköping to Söderköping, when the schedule is predicted by target values.

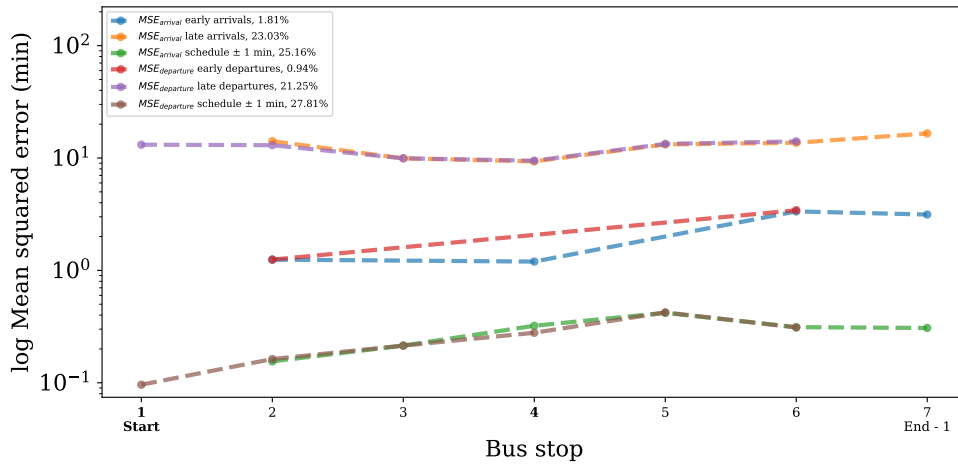


Figure 5.32: MSE on route 45 from Söderköping to Norrköping, when the schedule is predicted by target values.

Figure 5.33 depicts how the CPS, TBP and the LSTM model with Pang et al. [3] performs on route 45. The figure shows that the LSTM model have a lower median error for all stops and is able to produce more accurate predictions overall. This is supported by Table 5.8 which shows the evaluation metrics.

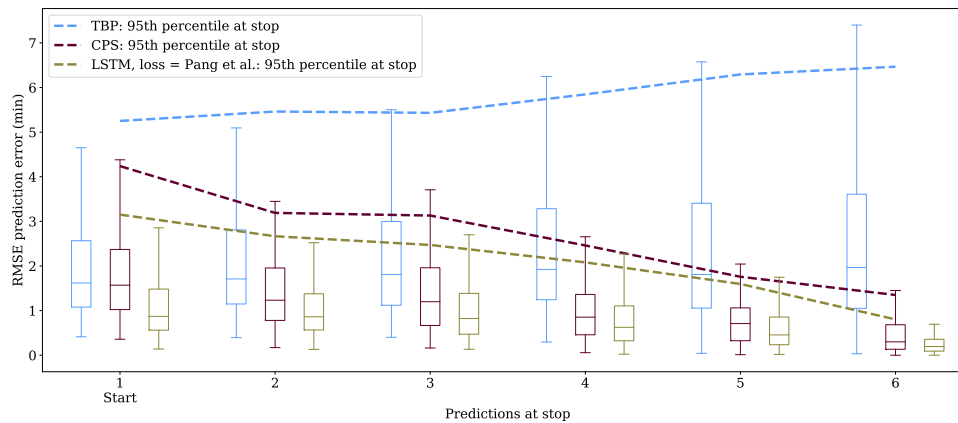


Figure 5.33: Prediction errors on the test set with 231 trips as measured in RMSE.

Method	Evaluation metrics					Prediction category (%)			
	Minutes				%	Correct	Under-estimated	Over-estimated	
	RMSE	RMSE*	MAE	MAE*	MAPE				
TBP	2.34 ± 1.83	2.08 ± 1.93	2.22 ± 1.83	1.38 ± 1.73	12.47 ± 7.25	33.6	48.9	17.5	
CPS	1.17 ± 0.5	0.82 ± 0.54	1.07 ± 0.47	0.42 ± 0.38	7.02 ± 2.62	54.4	21.1	24.5	
LSTM	MSE	0.85 ± 0.51	0.49 ± 0.55	0.77 ± 0.46	0.23 ± 0.35	5.04 ± 2.49	67.6	16.1	16.3
	Pang	0.84 ± 0.52	0.48 ± 0.55	0.76 ± 0.47	0.23 ± 0.35	4.78 ± 2.35	69.8	17.7	12.5
	U	1.04 ± 0.65	0.70 ± 0.71	0.93 ± 0.59	0.36 ± 0.47	5.51 ± 2.68	61.0	35.3	3.7
	MAE*	0.97 ± 0.51	0.60 ± 0.54	0.89 ± 0.46	0.28 ± 0.34	6.63 ± 3.34	59.8	11.0	29.2
	O	0.94 ± 0.53	0.58 ± 0.57	0.86 ± 0.47	0.28 ± 0.37	5.52 ± 2.76	63.4	9.8	26.7
LSTM Improvements	TBP	64.6%	76.9%	65.8%	83.3%	61.7%	107.7%	-	-
	CPS	26.3%	41.5%	29.0%	45.2%	31.9%	28.3%	-	-

Table 5.8: Evaluation results for route 45, $N = 231$.

The distribution prediction categories per stop for CPS and the LSTM model with Pang et al. [3] as loss function is illustrated by Figure 5.34. This time, the LSTM model is able to produce more predictions considered to be correct than the CPS for all stops.

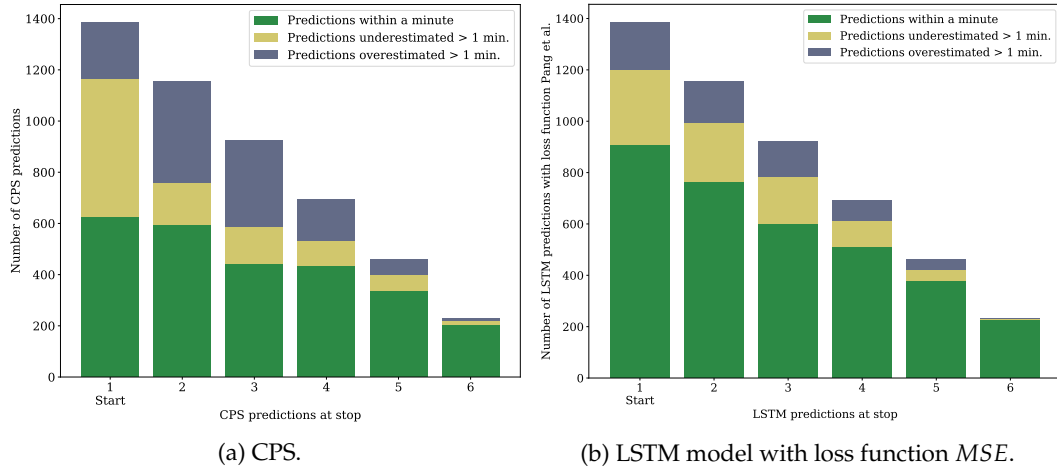


Figure 5.34: Type of error at each stop for the CPS and the LSTM model with loss function Pang et al. on route 45.

5.2 Summary

Thus far, the results for a set of eight hand-picked routes have been presented in further detail. The idea is that the selected routes should represent a wide selection of the different characteristics of all routes. This section aims to give a better overview of the result and present and present the improvements in a compact form. Table 5.9 shows a summary of the results. The LSTM method refers to the LSTM model with the lowest error in the metric *MAE* for each route.

Evaluation metric	Method	Route							
		70	303	616	20	119	3	450	45
MAE (min.)	TBP	2.55 ± 1.46	1.69 ± 1.04	2.32 ± 1.48	1.22 ± 1.12	2.21 ± 1.85	1.72 ± 1.26	2.88 ± 2.18	2.22 ± 1.83
	CPS	1.14 ± 0.63	0.75 ± 0.31	0.69 ± 0.32	0.66 ± 0.40	0.77 ± 0.38	0.66 ± 0.30	1.00 ± 0.43	1.07 ± 0.47
	LSTM	0.89 ± 0.62	0.64 ± 0.32	0.64 ± 0.33	0.39 ± 0.21	0.67 ± 0.40	0.49 ± 0.25	0.92 ± 0.52	0.76 ± 0.47
LSTM Improvements	TBP	65.1%	62.1%	72.4%	68.0%	69.7%	71.5%	68.1%	65.8%
	CPS	21.9%	14.7%	7.2%	40.9%	13.0%	25.8%	8.0%	29.0%

Table 5.9: Summary of the results.



6 Discussion

In this chapter, the applied method and the results are discussed. The first section is an analysis of the results and the section elaborates on the outcome of the conducted experiments for the selected routes. The applied methodology is also discussed in terms of replicability, reliability and validity. Last but not least, a general discussion is included about how the results can be put into practice.

6.1 Analysis of the results

The first overview analysis can establish that model based on a RNN with LSTM units works well in this domain. Table 5.9 shows that the LSTM model improves the predictions on all routes investigated. The model is able to find patterns and learn from past experience to create useful arrival time predictions for a different set of routes. It is also clear that the timetable generally is a quite bad predictor in terms of the evaluation metrics applied in this thesis. The fear of giving out times that will allow the bus to visit stops earlier and thus, possibly leave passengers behind, causes the timetable to underestimate the arrival times quite a bit for some routes.

As seen in the figures illustrating the error type at each stop, the CPS usually predicts close to the distribution mean since the error types often are fairly evenly distributed. Recall that the CPS value is presented as the delay parameter in the GTFS-RT feed which represent the scheduled arrival or departure time plus the delay. There might be underlying functionality attached to the system producing the predictions that is not included in the GTFS-RT feed.

As expected, the TBPs are constantly biased towards underestimations on all routes considered. This means that the passengers are currently provided with two different types of predictions. Either the time is underestimated or an attempt to hit the exact time of arrival. Currently, there is no expected last arrival time presented to passengers. Only the expected (mean) arrival time is. It means that the passengers best bet is to use the prediction produced by the CPS since TBP underestimates the arrival time most of the time. However, the results shows that if the CPS prediction deviated more than one minute from the target value, the prediction is generally as likely to be underestimated as overestimated. That is exactly why the LSTM model with loss function O is relevant. The results show that such a model is able to steer the errors towards the desired error category and predict a time that seldom is an

underestimation. Passengers can thus more easily make up plans if they have this worst-case arrival time available.

The first set of predictions produced by the CPS is close to the TBP which has a negative impact on the measurement result. However, the trend over all tested routes is clear when the median per stop is considered. The LSTM model with loss functions *MSE* and Pang et al. [3] is able to get a lower median error on most of the stops, regardless of the evaluation method.

For route 70, Figure 5.1 and Figure 5.2 indicates that the timetable and the observations are not very well aligned and there are room for improvements, especially on the later sections of the route in both directions. The bus is late to more than half of the stops. However, Table 5.1 shows that both of the LSTM models with the loss functions *MSE* and Pang et al. [3] manages to produce better predictions than CPS. The number of correct (within one minute error) predictions is also increased from 52.4% to 63.7% while the errors are evenly distributed as for CPS. Furthermore, the loss functions *U* and *O* manages to keep the percentage in their respective error category low while the evaluation metrics are comparable to CPS, despite the fact that their main task is to avoid overestimations and underestimations, respectively. The errors have successfully been directed to the desired error category at the cost of less accurate predictions. However, the metrics in Table 5.1 shows that both *U* and *O* produces results similar to CPS. That trade-off turned out to be generally valid across all lines. The safety margin that *U* and *O* entails us at the expense of less accurate predictions.

The largest improvement in percentage over the CPS was made on route 20 with a 40.9% improvement. A large part of the explanation is the few number of stops together with the fact that the bus often is late in one of the directions, as illustrated by Figure 5.15. Figure 5.18b shows how the CPS are heavily disadvantaged by predicting close to the timetable on the first stop. However, the figure also shows that the LSTM model has a better median value for the rest of the stops indicating an overall improvement even if the very first stop had not been considered.

The smallest percentage improvement of 7.2% over the CPS was noted on route 616. This is quite a long route with relatively many stops where the bus often is late. Even though the improvement is small compared to the CPS, the difference between the TBP and LSTM is the greatest over all routes tested. It indicated the need for a prediction system in order to provide passengers with a good service.

Route number 616 and 450 are considered to be of the route type inter-city. Table 5.9 shows that these routes were the hardest to improve compared to the CPS, with an 7.2% and 8.0% improvement, respectively. The inter-city express routes on the other hand enjoyed improvements of 21.9% on route 70 and 29.0% on route 45. The improvements over the CPS on the city routes were quite spread. The short city route number 20 in Linköping can be improved with as much as 40.9% over the CPS, while route number 119 in Norrköping was improved by 13.0%.

The analysis is based on data from a period in which it is winter. Slightly tougher road conditions due to weather may partly contribute to the amount of delayed buses. It is therefore not unimaginable that for instance, the TBP has a less average error during the summer.

6.2 Loss functions

An interesting observation is that the loss function *MAE** makes the model inconsistent. The evaluation metrics is generally worse than both *MSE* and Pang et al. [3] and the prediction category distribution indicates that the predictions are either correct or underestimated. *MAE** is, in the light of *MSE* and Pang et al [3], more or less producing nonsense results. One possible explanation for this behavior is the lack of feedback difference in the interval between plus minus one minute. The relaxation of the acceptable region seems to make the network confused or lazy rather than simplifying. It is perhaps not to surprising since the net-

work is suddenly not given any incentives to strive for improvements when the prediction is within the desired region.

6.3 Method

The parameters for training the models suggested in this thesis relies heavily upon the literature, as presented in Chapter 2. The only training parameters that have been empirically tuned is the learning rate, patience and the constants in the loss functions U and O . Regarding the training parameter values, these are not claimed to be optimal and more work can be put into finding optimal training parameter values.

The structure of the LSTM RNNs used as a basis for the models is also based on the literature, and the work conducted by Pang et al. [3]. Although the choice of network structure and dropout is based on related work, it is interesting to continue exploring alternatives and their impact on the result.

Using only complete sequences up to the very last stop means that the last stop on each route is never evaluated. An alternative approach would be to include these sequences and perhaps all complete sequences and evaluate them against corresponding sequences. The data source would arguably be utilized more efficiently and more training sequences would be available. However, there are only a few complete sequences available in the dataset which reduces the impact of the chosen method.

An alternative to only consider predictions in conjunction with visiting bus stops is to make predictions more often, i.e. between stops as well. The positions are updated with 1 Hz and longer sequences can thus be constructed. Routes with long distances between stops would benefit from detecting anomalies earlier. However, this is not how the CPS operates and the models were thus adapted to today's system. The approach in this thesis is easily extendible to include more frequent predictions. In such case there might be good to know that there is an upper limit to the length of the sequences where the gradient might have a difficult time propagating properly without vanishing or exploding.

6.4 Applicability

The pervasive theme in this thesis is to solve a very practical problem. The goal is to provide passengers and passengers-to-be with as accurate information as possible in order to provide the best possible service. By studying bus lines over time, the results show that it is possible to extract patterns and dependencies that can successfully be used to predict the time of arrival several stops ahead. When a model for a line is trained, predictions can quickly be made using the best found weights. Also, the two use cases studied in this thesis captures two real needs that passengers and passengers-to-be have and that is not being met today.

Improved arrival time predictions on average is part of increasing the attractiveness of public transportation in general, together with other important aspects such as offering shorter travel times and more frequent trips. An overall better service can attract new and more passengers which in turn leads to increased revenue. Increased revenue makes it possible to continue with investments in the service with ultimately leads to a better public transportation system. There are accurate predictions about the time of arrival part of being able to give passengers a more attractive service.

From the driver's perspective, predictions of arrival can certainly contribute to increased stress if the times are perceived as a fact more than an estimate. Knowing the outcome of how other, perhaps more experienced drivers have driven in similar situation before might put some extra pressure on keeping the same pace. There is also a monitoring aspect to take into consideration. Every decision taken by the driver has consequences that are not only systematically monitored and saved, but also actively used by the prediction system to provide passengers with information. Also, different drivers probably have different driving charac-

teristics affecting the arrival time. It is conceivable that, for instance, an anonymous driver ID can capture these differences and for that reason is included in future, similar models. That requires another discussion about what it is likely to have for consequences for ethical and moral concerns.

An open question is how the models should handle new data. The world is a place of constant change and some sort of robust online learning would be needed in order to keep the models up to date in the long term. Such a solution should also consider the computational power needed and find a computationally efficient way of keeping the models up-to-date.



7 Conclusion

This chapter concludes the thesis and addresses answers to the studied research questions. The research questions will be addressed and the goal in this thesis will be evaluated. Furthermore, a section with future work is presented along with suggestions to further improve the proposed models.

First, an everyday problem was identified. ÖT uses a system to predict arrival time for buses which is presented to the passengers and passengers-to-be. It was unclear how well this system performed and there was a need to investigate if it was possible to improve the arrival time predictions on certain lines to ultimately facilitate the everyday lives for people affected. Second, two types of passengers with different needs were identified for which models were designed in order to meet their respective needs. Third, the result section showed that models based on RNNs with LSTM units are able to produce better results in all measurements compared to the existing approach. Now only a lower error on average is achieved, but also a higher proportion of the total predictions that are classified as correct on all routes tested. Furthermore, two new ways of predicting the arrival time is presented. By selecting a loss function that penalizes underestimations or overestimations more harshly, the results shows a trade-off between precision and certainty that the predictions in most cases does not fall into the unwanted prediction error category. With this in mind, the research questions below can be answered.

1. How well can a recurrent neural network model predict arrival times compared to the existing prediction system used by Östgötatrafiken?

Answer: The results show that models based on recurrent neural networks can produce more accurate arrival time predictions than the existing prediction system on all routes tested. The LSTM model with the lowest median error on each route showed improvements ranging from 7.2% to 40.9% compared to the existing prediction system used by Östgötatrafiken today.

2. The current passenger and the passenger-to-be have different needs in terms of predictions. How can this be facilitated?

Answer: One way to facilitate their needs as shown in the results is to train a recurrent neural network model with a suitable loss function for each need. Using loss function U , we show that the predicted arrival times are overestimated in between 0.8% to 3.8%

of the times on all routes tested. If we instead use the loss function O , we show that the arrival times were underestimated between 3.1% to 9.8% of the times. The remainder of the predictions are either within one minute from the target value or an underestimation (U) or overestimation (O).

3. What cost functions and validations are suitable for arrival time predictions?

Answer: The results show that the loss function proposed by Pang et al. [3] has the lowest mean error in virtually all evaluation metrics. Furthermore, a loss function that weights the errors depending on the type is suitable to address the identified needs. The functionality was validated by running tests and then observe and compare the predictions with the observations.

The answers to the research questions indicates that we did in fact manage to reach our goal to study how better arrival time predictions can be made and for whom. Two cases were identified with different needs. The current passenger and the passenger-to-be. Their needs can be addressed by selecting a non-symmetrical loss function that punished either underestimations or overestimations more harshly. Also, the LSTM model with loss function Pang et al. [3] can reduce the mean prediction error and produce more predictions within one minute from the target time on all routes tested. Because the solution is relatively easy to expand with more data or other data sources, we have a good basis for building and applying our findings out in the real world.

7.1 Future work

The final section of this thesis will bring up interesting suggestions for potential improvements and alternative approaches or solution in general. The following list is subject for future work as it could possibly improve the proposed models:

- Perhaps one of the more interesting ideas is to shift focus from the standard regression approach trying to predict the exact time to predicting an interval. Predicting the confidence interval would have the great benefit of also providing a certainty value along with the predicted arrival time. An interesting alternative can be to predict the parameters of a distribution which would facilitate the interpretability and make the model more rigorous. In such case, the parameters for the loss functions U and O could be calculated directly from the distribution which could replace both models with one.
- The dataset could be used more efficiently. One idea is to interpolate missing values in the `TripUpdates` feed with AVL data from `VehiclePositions` to construct complete sequences of arrival and departure times at stop. That idea could be developed further to construct sequences longer than the number of stops and one-hot encode the type of the segment. Suitable classes could be for instance intersection, bus stop, or normal road. The prediction system would then predict all segments to move the state forward but only publish the predictions relevant to the passengers.
- Each bus has a predefined schedule which can include several routes. It would therefore be interesting to associate predictions with buses and include the entire trajectory, not only the parts which is associated with a trip instance. In that way, information about past events from other trips such as delays could then be conveyed faster to passengers.
- A closer interval between the predictions would propagate information to the passengers at a faster pace. Since a new set of predictions is made in connection with a bus stop visit, it might take a while before events become available. It is especially a problem on lines with long distances between two adjacent stops. Updating the predictions more often by dividing the lines into smaller sections would solve that particular problem and the effect would be interesting to study further.

- The parameters in the loss functions designed to avoid underestimating and overestimating the arrival times is an interesting design parameter. Instead of empirically finding suitable constants, it would be interesting to go the other way around and calculate the values based on the desired probabilities for avoiding underestimations and overestimations.
- The training parameters is heavily based upon the literature. Although the models in this thesis is not claimed to be optimal, it would be interesting to further investigate parameter optimization and study different network architectures, like stacked LSTM models, for instance.
- Online learning to keep the models up-to-date so that they do not lose their accuracy over time.



Bibliography

- [1] *Transport Analysis Statistics*. https://www.trafa.se/en/public-transport-and-publicly-financed-travel/regional_scheduled_public_transport/. Accessed: October 2019.
- [2] Bin Yu, Huaizhu Wang, Wenxuan Shan, and Baozhen Yao. "Prediction of Bus Travel Time Using Random Forests Based on Near Neighbors". In: *Computer-Aided Civil and Infrastructure Engineering* 33.4 (2018), pp. 333–350.
- [3] Junbiao Pang, Jing Huang, Yong Du, Haitao Yu, Qingming Huang, and Baocai Yin. "Learning to Predict Bus Arrival Time From Heterogeneous Measurements via Recurrent Neural Network". In: *IEEE Transactions on Intelligent Transportation Systems* (2018).
- [4] Peilan He, Guiyuan Jiang, Siew-Kei Lam, and Dehua Tang. "Travel-Time Prediction of Bus Journey With Multiple Bus Trips". In: *IEEE Transactions on Intelligent Transportation Systems* (2018).
- [5] *Interactive map for vehicle positions in real-time at Östgötatrafiken*. <https://www.ostgotatrafiken.se/reseinfo/planera-din-resa/Karta/>. Accessed: October 2019.
- [6] Saskia Sassen. *Cities in a world economy*. Sage Publications, 2018.
- [7] Roberta Guglielmetti Mugion, Martina Toni, Hendry Raharjo, Laura Di Pietro, and Samuel Petros Sebatu. "Does the service quality of urban public transport enhance sustainable mobility?" In: *Journal of Cleaner Production* 174 (2018), pp. 1566–1587.
- [8] Luigi Dell’Olio, Angel Ibeas, and Patricia Cecin. "The quality of service desired by public transport users". In: *Transport Policy* 18.1 (2011), pp. 217–227.
- [9] Bin Yu, William HK Lam, and Mei Lam Tam. "Bus arrival time prediction at bus stop with multiple routes". In: *Transportation Research Part C: Emerging Technologies* 19.6 (2011), pp. 1157–1170.
- [10] Mark A Aizerman. "Theoretical foundations of the potential function method in pattern recognition learning". In: *Automation and remote control* 25 (1964), pp. 821–837.
- [11] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*. ACM. 1992, pp. 144–152.
- [12] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. "Kernel methods in machine learning". In: *The annals of statistics* (2008), pp. 1171–1220.

- [13] Pengfei Zhou, Yuanqing Zheng, and Mo Li. "How long to wait?: predicting bus arrival time with mobile phone based participatory sensing". In: *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM. 2012, pp. 379–392.
- [14] Jun Gong, Mingyue Liu, and Sen Zhang. "Hybrid dynamic prediction model of bus arrival time based on weighted of historical and real-time GPS data". In: *Control and Decision Conference (CCDC), 2013 25th Chinese*. IEEE. 2013, pp. 972–976.
- [15] Moshe Ben-Akiva, Michel Bierlaire, Haris Koutsopoulos, and Rabi Mishalani. "DynaMIT: a simulation-based system for traffic prediction". In: *DACCORD Short Term Forecasting Workshop*. Delft The Netherlands. 1998, pp. 1–12.
- [16] Li Li, Xiaonan Su, Yi Zhang, Yuetong Lin, and Zhiheng Li. "Trend modeling for traffic time series analysis: An integrated study". In: *IEEE Transactions on Intelligent Transportation Systems* 16.6 (2015), pp. 3430–3439.
- [17] Steven I-Jy Chien, Yuqing Ding, and Chienhung Wei. "Dynamic bus arrival time prediction with artificial neural networks". In: *Journal of Transportation Engineering* 128.5 (2002), pp. 429–438.
- [18] Ranhee Jeong and R Rilett. "Bus arrival time prediction using artificial neural network model". In: *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No. 04TH8749)*. IEEE. 2004, pp. 988–993.
- [19] Chi-Hua Chen. "An arrival time prediction method for bus system". In: *IEEE Internet of Things Journal* 5.5 (2018), pp. 4231–4232.
- [20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [21] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.
- [22] Norman R Draper and Harry Smith. *Applied regression analysis*. Vol. 326. John Wiley & Sons, 1998.
- [23] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407.
- [24] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), p. 533.
- [25] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).
- [26] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization". In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159.
- [27] Tijmen Tieleman and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.
- [28] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. "On the convergence of adam and beyond". In: *arXiv preprint arXiv:1904.09237* (2019).
- [29] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.
- [30] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 4960–4964.
- [31] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE. 2013, pp. 6645–6649.

- [32] Vivek Veeriah, Naifan Zhuang, and Guo-Jun Qi. "Differential recurrent neural networks for action recognition". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4041–4049.
- [33] Yongxue Tian and Li Pan. "Predicting short-term traffic flow by long short-term memory recurrent neural network". In: *2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)*. IEEE. 2015, pp. 153–158.
- [34] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.
- [35] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.
- [36] Alex Graves. "Supervised sequence labelling". In: *Supervised sequence labelling with recurrent neural networks*. Springer, 2012, pp. 5–13.
- [37] Paul J Werbos et al. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.
- [38] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).
- [39] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM". In: (1999).
- [40] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep contextualized word representations". In: *arXiv preprint arXiv:1802.05365* (2018).
- [41] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.
- [42] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feed-forward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.
- [43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.
- [44] Yarin Gal and Zoubin Ghahramani. "A theoretically grounded application of dropout in recurrent neural networks". In: *Advances in neural information processing systems*. 2016, pp. 1019–1027.
- [45] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.
- [46] Mattias Tiger and Fredrik Heintz. "Online sparse Gaussian process regression for trajectory modeling". In: *Information Fusion (Fusion), 2015 18th International Conference on*. IEEE. 2015, pp. 782–791.
- [47] Mattias Tiger and Fredrik Heintz. "Gaussian Process Based Motion Pattern Recognition with Sequential Local Models". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2018, pp. 1143–1149.
- [48] Carl Edward Rasmussen. "Gaussian processes in machine learning". In: *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71.

- [49] Yarin Gal and Zoubin Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*. 2016, pp. 1050–1059.
- [50] Cort J Willmott and Kenji Matsuura. “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance”. In: *Climate research* 30.1 (2005), pp. 79–82.
- [51] Tianfeng Chai and Roland R Draxler. “Root mean square error (RMSE) or mean absolute error (MAE)?—Arguments against avoiding RMSE in the literature”. In: *Geoscientific model development* 7.3 (2014), pp. 1247–1250.
- [52] Arnaud De Myttenaere, Boris Golden, Bénédicte Le Grand, and Fabrice Rossi. “Mean absolute percentage error for regression models”. In: *Neurocomputing* 192 (2016), pp. 38–48.
- [53] NOPTIS. <http://www.noptis.org/cmlv/default.asp>. Accessed: October 2019.
- [54] Trafiklab. <https://www.trafiklab.se/>. Accessed: October 2019.
- [55] *The GTFS Realtime specification*. <https://developers.google.com/transit/gtfs-realtime/>. Accessed: October 2019.
- [56] *The GTFS specification*. <https://developers.google.com/transit/gtfs/>. Accessed: October 2019.
- [57] *The protocol buffer file format documentation overview*. <https://developers.google.com/protocol-buffers/docs/overview>. Accessed: October 2019.
- [58] *Github project gtfs-realtime-bindings*. <https://github.com/MobilityData/gtfs-realtime-bindings>. Accessed: October 2019.
- [59] Tianle Chen, Brian Keng, and Javier Moreno. “Multivariate Arrival Times with Recurrent Neural Networks for Personalized Demand Forecasting”. In: *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2018, pp. 810–819.
- [60] Dominic Masters and Carlo Luschi. “Revisiting small batch training for deep neural networks”. In: *arXiv preprint arXiv:1804.07612* (2018).
- [61] Yoshua Bengio. “Practical recommendations for gradient-based training of deep architectures”. In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.
- [62] François Chollet et al. *Keras*. <https://keras.io>. Accessed: October 2019. 2015.
- [63] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. “TensorFlow: A system for large-scale machine learning”. In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pp. 265–283. URL: <https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf>.