NTNU – Trondheim
Norwegian University of
Science and Technology

On Implementations of Bus Travel Time Prediction Utilizing Methods in Artificial Intelligence

Erlend Dahl
Aleksander Aas Sjåfjell
Simen Skogen

Master of Science in Computer Science
Submission date: June 2014
Supervisor: Jo Skjermo, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

On Implementations of Bus Travel Time Prediction Utilizing Methods in Artificial Intelligence

Aleksander Sjåfjell, Erlend Dahl and Simen Skogen

Abstract—Travel time prediction is an important part of intelligent transportation systems. This work is a continuation of a state-of-the-art review identifying four prominent machine learning algorithms used for bus arrival prediction: Kalman filtering, k-Nearest Neighbor, Artificial Neural Network, and Support Vector Regression. The Weka library implementations of the latter three classifiers are utilized on bus travel time prediction in Trondheim, Norway. A brief overview of the data collection process and a rationale for selecting the data sources is given. Next, the parameters of the classifiers are optimized, and different partitions of training and test periods are evaluated. An attribute analysis investigates the use of other variables hypothesized to influence travel time prediction, such as weather, to improve the prediction accuracy. Finally, a proof of concept model for real-time prediction is presented, with its performance being competitive to the existing real-time system.

Index Terms—intelligent transportation system, prediction, support vector machine, k-nearest neighbor, artificial neural network, machine learning, bus arrival time prediction

1 Introduction

Intelligent transportation systems (ITS) is a generic term for the integrated application of communications, control and information processing technologies to transportation systems [1]. This can provide improved information services, simplified management, and potentially smoother traffic flow [2] [3]. Since the early 2000s the advancements in information and communication technologies have made it possible for transport systems to become more intelligent, efficient, safe and eco-friendly. Car navigation, automatic vehicle location, inductive loop detection, variable message signs, and speed cameras are examples of early ITS technologies that are already extensively used worldwide [4] [5] [6].

Machine learning, a branch of Artificial Intelligence (AI), and statistics have been applied to ITS-collected data in a progressively increasing extent. Particularly applications that require large amounts of recent data, such as travel time prediction, has become a vital and prominent part of AI research.

This work is based on a state-of-the-art literature review of travel time prediction for public transport in urban areas where AI methods were utilized, see Appendix I. An extended version of this work can be found at [7]. The literature review identified and compared four well established methods for travel time predictions in the traffic domain, namely k-Nearest Neighbor (kNN), Artificial Neural Network (ANN), Support Vector Regression (SVR) and Kalman filtering.

The literature review was unable to pinpoint a single classifier as the best option for bus travel time prediction. Therefore, this research will apply and compare these classifiers on bus travel time prediction in the medium sized city of Trondheim, Norway. Several aspects of the prediction are considered in the research:

- Fetching, merging and preprocessing data from multiple data sources (Chapter 2).

. This work was partially funded by the SMIO-project, a research project funded by the Regional Research Fund Capital area.

- Optimizing the method-specific parameters for higher prediction accuracy (Chapter 3).
- Comparing the methods' performance on a wide range of different dataset types to decide which datasets are most suitable, and try to identify the best method (Chapter 4).
- Analyzing the influence of different attributes in the datasets to identify the most important information (Chapter 5).

In chapters 2-5, the data is aggregated on individual trips, i.e. one data row for each bus arrival at the chosen bus stop. Consequently, predictions on future buses may be done after the most recent bus has arrived. This prediction is typically 15-30 minutes into the future, depending on the headway between the buses. In Chapter 6 however, data is aggregated on every bus stop passage. This corresponds to a near real-time prediction, as new predictions may be done at every bus stop passage.

Furthermore, Chapters 2-5 consider prediction of arrival delay. For reasons to be discussed, Chapter 6 considers prediction of travel time instead of delay.

Chapter 2 explains how data was fetched from different sources and combined into a final data table to prepare for prediction. Chapter 3 explains how the three methods (kNN, ANN and SVR) are optimized for the datasets, and Chapters 4 and 5 explains the dataset analysis that compares the methods on the different dataset types, and the attribute analysis that attempts to find out which attributes works best on the different dataset types. In Chapter 6, a proof of concept method for real-time prediction is presented and compared to the existing system. Finally, in chapters 7 and 8, conclusions and future work are given. Additional details on the methods, datasets and analyses are added as appendices.

2 Dataset preparations

AI methods, such as kNN, ANN and SVR (often referred to as classifiers), require a training set to create their models and a testing set for performance validation. The sets contain instances, or rows, that hold values for different attributes, very much like a database table. Training and testing is normally carried out with k-fold cross-validation; the data is randomly partitioned into k equal subsets, with k-1 subsets for training and the remaining subset for testing. By repeating the process k times, every instance is used both for training and testing, resulting in a legitimate evaluation of how the methods will generalize to unseen data. However, when predicting travel time based on earlier data, partitioning into random subsets should be avoided, since the only interesting timespan to predict in practice is the future. Consequently, in this work the datasets were separated in such a way that the instances of the test set follow the training instances chronologically, and the relative sizes of the partitioned training and testing sets varies from dataset to dataset.

Trips from bus line 8 (going from *Kongens Gate K2* to *Blakli*), as illustrated in Figure 1, were set as basis for predictions. Its total length of 8140 meters include 20 bus stops, 16 intersections (of which 13 are signalized), 11 pedestrian crossings, two roundabouts and a football stadium, located at *Lerkendal*. It is also among the most frequently driven lines in Trondheim. Due to the high frequency of intersections, crossings and stops, travel time becomes hard to predict. However, this is suitable for parameter optimization and attribute analysis, as different configurations of attributes and parameters will affect the predictions to such an extent that an optimal configuration is more likely to be found.

In order to test the classifiers in different conditions, three bus stops with different levels of occupancy along the route were selected for further analysis: Samfundet (near the city center of Trondheim), Lerkendal (close to the football stadium) and Steindalsvegen (in the outskirts of the city).

2.1 Data sources

Atb AS (AtB) is the administrative company for most of the public transport in Trondheim, Norway. AtB provided multiple tables with logging information from their bus operations, including every time a bus passes bus stops and virtual loops, with information such as

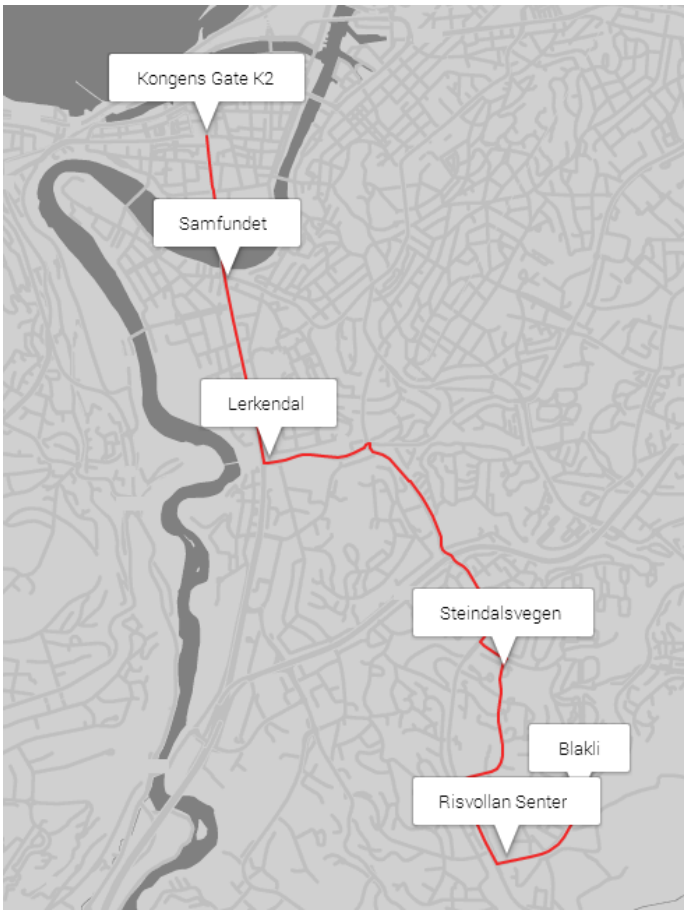

Fig. 1: Route *Kongens Gate K2 - Blakli*, line 8. Map by Google Maps

the specific vehicle in use, which route it was driving, the scheduled and actual time of passing and the duration of the stop at this location. The tables also contained static information about all bus stops and all bus routes in Trondheim, which proved useful for selecting a suitable research area.

In addition to the bus logs, AtB also provided anonymized ticket data. This included logs from smartphone tickets, manual cash tickets, and for some periods travel card validations. In Trondheim, all tickets must be validated when entering the bus, even if using a monthly subscription. This means that the ticket data may be an important factor for bus stop-time, since each ticket validation results in a minor delay. The travel card validations were used in the final datasets, but the smartphone and cash ticket data was excluded, as it was aggregated on a daily basis, and therefore difficult to combine with the other data sources. A passenger counting system was installed in one of the buses on the route at the time of writing. Counting data from this bus was included as an attribute in the datasets.

The shifting weather conditions in Trondheim were hypothesized to influence the number of people traveling by bus. Hence, weather data was collected from yr.no, which is a joint online weather service provided by the Norwegian Meterological Institute and the Norwegian Broadcasting Corporation [8]. Data collected included temperature, precipitation, wind and humidity.

Another assumption was that people attending large events cause more traffic, which in turn influences the bus delays. In Trondheim, the largest recurring events are the football matches at Lerkendal stadium, that often cause traffic jams in the proximity of the stadium. Attendance statistics were collected from TV2, a Norwegian television broadcaster.

The Norwegian Public Road Administration (NPRA) provides information on travel time for arterial roads in Trondheim. This information is based on passage detections of vehicles with Autopass transponders installed [9]. Due to a large extent of bus lanes in the city, the private transport is more or less separated from the public transport, and the correlation of the respective travel times are hypothesized to decrease where bus lanes are present. Also, since the arterial roads are not part of the areas of interest for bus arrival prediction, the data obtained from NPRA was not utilized.

HTML scraping was applied to regularly retrieve bus arrival predictions from the current real-time service available at AtB's website. These predictions were applied for comparison to the proof of concept model, as presented in Chapter 6.

All data sources were inserted into an SQL database, which was then accessed locally on the development computers to provide the performance needed to run the analysis. The weather data had to be interpolated to fill a few missing data points . A more detailed explanation of how the data sources were combined into a single table can be found in Appendix A.

Table 1: The full list of datasets.

ID	Description	Number of sets
1a	Train on one hour, test on the following hour.	30
1b	Train on two hours, test on the following hour.	30
1c	Train on three hours, test on the following hour.	30
1d	Train on four hours, test on the following hour.	30
1e	Train on five hours, test on the following hour.	30
2	Train on Monday-Thursday, test on Friday.	12
3	Train on Monday-Sunday, test on each of the days the following week.	42
4	Train on Monday-Friday, test on each of the weekdays the following week.	30
5	Train on two weeks, test on the following week.	6
6	Train on three weeks, test on the following week.	12
7	Train on three weeks, test on the week one week later.	6
8	Train on three weeks, test on the week two weeks later.	6
9	Train on one day, test on the same day the following week.	21
10	Train on two consecutive recurring days, test on the same day the following week.	21
11	Train on three consecutive recurring days, test on the same day the following week.	21
12	Train on four consecutive recurring days, test on the same day the following week.	21
13	Train on five consecutive recurring days, test on the same day the following week.	21
14	Train on one month, test on the following month.	3
15	Train on one month, test on each of the weeks in the following month.	12
16	Train on two succeeding months, test on the following month.	3
17	Train on two succeeding months, test on each of the weeks in the following month.	12
18a	Train on 100 succeeding days, test on the 1, 2, 7 and 14 following days.	12
18b	Train on 100 succeeding days, test on 1, 2, 7 and 14 succeeding days approximately three months later.	12
Total		423

2.2 Dataset assembly

423 different datasets were assembled from the data, 141 for each of the three bus stops. The exact attributes of the final datasets are listed in Appendix B, Table A1. The datasets were mostly from the period 01.09.2013 to 23.12.2013, but there are a few testing sets from February 2014. These periods were chosen because they had data without any missing data points (for example bus routes with missing parts due to system failures) or extremely high or low values on any of the selected bus stops. The annual autumn vacation for schools was avoided as much as possible, as it was expected to change the traffic pattern from regular day-to-day operations. When all school buses are stopped, and random vacation activities take their place, the traffic patterns will be different.

This range of datasets was specifically assembled to test a range of different cases:

1) **Training length:** some datasets had training sets with a duration of a few hours, others multiple days, weeks, or even months.
2) **Day patterns:** some training sets were from consecutive days within the same week, others from the same days in consecutive weeks (for example five Mondays in a row).
3) **Continuity:** some test sets continued right where the training set stopped, while others started weeks later.

An exact list of the datasets can be seen in Table 1, where they are listed in different groups depending on where they fit in the aforementioned cases. Note that each group has at least one dataset for each of the three bus stops, and then most of them have additional datasets of the same type, but from different weeks or months to verify the results. Because of the ticket data, these datasets can not be publicized without permission from AtB.

2.3 Representative datasets

A representative subset of the datasets was created for the more time consuming analyses, since running all tests on all 423 datasets would be infeasible because of budget restrictions. As

the datasets were created to test a wide range of assumptions, the representative subset should resemble a variety similar to the 423 datasets, in order to produce reasonable results. Therefore, one representative set was selected from each of the groups shown in Table 1, except set 18b, since the datasets in group 18 were very large and time-consuming to classify. These sets will be known as the 23 representative datasets for the rest of this work.

2.4 Classifier implementations

To easily test a large variation of classifiers and classifier parameters, it was decided to use the Weka library (v.3.6.10) [10]. This allowed easy setup in Java, and no time was spent reimplementing AI methods. As Weka did not support Kalman filters at the time of writing, the Kalman filter was excluded from the analyses.

2.5 Processing power

The parameter analyses were particularly time-consuming and the processing power of three regular desktop computers was limited. Therefore, two Amazon Elastic Compute Cloud (Amazon EC2) computers with 32 cores each was rented for a week to supplement the processing power of the desktop computers [11].

2.6 Rating function

It is important to recall that travel time prediction systems for buses is created to aid the users deciding which bus to take, and when to go to the bus stop. The mean absolute error (MAE) gives the average prediction error in seconds, while the standard deviation of the prediction errors (STD) indicates the stability of the predictions. Even though a low MAE is important, it is just a part of what the users want. As a user, the stability of the predictions is often far more important, since this is what shows how trustworthy the system is. It is adequat to know that the bus is 5 minutes late as long as this prediction is stable and does not abruptly change. A stable configuration is particularly important in this work, since the parameter optimizations were only run on a subset of all the datasets.

A rating function taking this into account was created to decide which configuration of attributes or parameters that performed best. A good function is one that rewards systems with a low MAE and STD at the same time, and it was therefore decided to be $Rating = MAE + STD$, where lower is better. As there is no standards or good literature on weighing such a function now, the weights on both parameters are simply 1. This is good enough for our analysis, but other weights should be considered for practical use.

3 Parameter optimization

kNN, ANN and SVR are all data dependent methods. Since there is no general rule on how to structure the methods for a specific problem, the optimal parameters are typically found empirically, rather than analytically [5] [12] [13]. Consequently, method specific parameters should be optimized on individual datasets in order to classify with maximum accuracy. However, in this chapter, each classifier was optimized for multiple datasets with identical attributes, but different time structures, to assess in which degree an optimization could be generalized.

Before the different classifiers were optimized, each of them was run with their default parameters on all the 423 datasets. These results are compared with the optimized results in Section 3.4 to evaluate the parameter optimization.

3.1 k-Nearest Neighbor parameters

The kNN implementation in Weka uses cross-validation and distance weighting [14]. It has three important parameters: the distance measurement, how many neighbors to use (k), and the algorithm used for finding the k nearest neighbors. The algorithms for finding the nearest neighbors also had some parameters of their own which had to be tested. To save computing time, only the representative datasets were used, as discussed in Section 2.3.

There were four different neighbor algorithms implemented: BallTree, CoverTree,

KDTree, and LinearNN [15] [16] [17]. Some manual testing indicated that most of the parameters on the individual neighbor algorithms made no differences on the datasets used in this analysis. The only setting that changed the predictions was using different distance functions on LinearNN. The three different distance algorithms available were Chebyshev distance (the greatest distance from all dimensions), Manhattan distance (sum of the distances in all dimensions), and Euclidean distance (direct line-distance). See Figure 2 for a visual explanation. This left six different neighbor algorithm configurations to test: BallTree, CoverTree, KDTree, LinearNN with Manhattan distance, LinearNN with Chebyshev distance, and LinearNN with Euclidean distance.

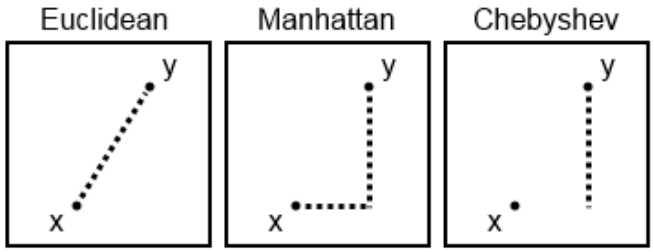

Fig. 2: The different distance algorithms of kNN: Euclidean (shortest path), Manhattan (sum of distance in x- and y-dimension) and Chebyshev distance (largest of distances in x- and y-dimension). Here shown between two points in a 2D space.

For distance weighting on the classifier itself, Weka has three alternatives: No distance weighting (called "None"), weight by $1/d$ (called "InverseDistance"), and weight by $1-d$ (called "Distance"). d is the distance between the current instance and its neighboring instances, used for selecting the nearest neigbors.

For each of the representative datasets, the kNN classifier was run with all combinations of the six neighbor algorithm configurations, the three distance weighting options, and a total of 185 different k-values. For a detailed overview of the k-values, see Appendix C.1.

The results of the tests were given a rating as explained in Section 2.6. As shown in Table 2, all of the five best configurations used the LinearNN-algorithm with Manhattan-distance for choosing neighbors, and the InverseDistance method for weighting the distances. In fact, the top 15 configurations only differ in k-value, where all of them use values around $n/25$ (where n is the number of instances in the dataset). As can be seen, there are very small differences in rating among the best configurations, but $n/22$ is slightly better than the neighboring k-values.

The results indicated that the optimal kNN parameters were the LinearNN algorithm with Manhattan-distance as nearest-neighbor algorithm, InverseDistance as distance weighting, and a k-value of $n/22$ (or 5 if this value was lower than 5, as discussed in Appendix C.1).

3.2 Artificial Neural Network parameters

Weka's implementation of a neural network is a feed forward artificial neural network that applies backpropagation to classify instances [10].

There are four parameters of the ANN that are essential to optimize in Weka: the network topology, learning rate, momentum and epochs (training time). As ANN has a much higher run time than kNN, it is not feasible to test all combinations of these parameters within a reasonable time frame. Tests were therefore first executed on the hidden layers, then learning rate and momentum, and finally the number of epochs.

The topology of an ANN is an explicit representation of what the network has learned in the training phase. Too few nodes will give the network problems with representing data learned, while too many nodes will lead to a network where the nodes act like a pure memory unit and generalization is lost.

There are rule-of-thumb methods for finding the number of nodes in the hidden layers of a neural network [18] [19] [20]. A neural network with one or two hidden layers is often sufficient to solve non linear problems, but if long training time is acceptable and accuracy is very important, 3 layers can be used. [21]

See Appendix C.2, Table A2 for a detailed description of the testing of network topologies, and the selection of the five best of them.

The learning rate and momentum ranges between 0 and 1. The learning rate controls

Table 2: The 5 best configurations of the kNN classifier, and the 3 worst for comparison (note: the entire table consists of 73 260 rows, and can not be included for obvious reasons). n is the number of instances in a dataset.

Nearest-neighbor algorithm	Distance weighting	k-value	MAE	STD	Rating
LinearNN with Manhattan-distance	InverseDistance	n/22	103.95	44.25	148.20
LinearNN with Manhattan-distance	InverseDistance	n/23	103.90	44.34	148.24
LinearNN with Manhattan-distance	InverseDistance	n/21	104.06	44.21	148.26
LinearNN with Manhattan-distance	InverseDistance	n/28	104.37	43.90	148.27
LinearNN with Manhattan-distance	InverseDistance	n/27	104.39	43.89	148.27
...	...	...	...	...	...
LinearNN with Manhattan-distance	Distance	98	171.18	150.50	321.68
LinearNN with Manhattan-distance	Distance	99	171.20	150.50	321.69
LinearNN with Manhattan-distance	Distance	100	171.21	150.49	321.70

how fast the weights of the nodes are changed during the training phase [22]. If this value is close to 1, it may cause the network to learn quicker, but if the variability of the input data is high, the prediction accuracy may be reduced due to oscillations [23]. In order to increase the learning rate without leading to oscillations, the momentum is used to control the effect of past weight changes on the current direction of weight change [24]. A high momentum value can result in faster convergence, but the best solution may not be found [22]. The learning rate and momentum parameters were adjusted on the five best networks to find the optimal values, which then were used to find the optimal number of epochs for the same networks. For a more detailed description of the parameter value selection process, see Appendix C.

3.3 Support Vector Regression parameters

The SVR implementation in Weka has two important parameters: the C-value, and the kernel to use. The kernel itself may have additional parameters that need testing. Weka supports four kernels that were chosen for testing: polynomial, normalized polynomial, Pearson VII function based (PUK), and radial basis function (RBF). The rest of the kernels supported by Weka were excluded from these tests because they needed precomputed matrices or specially formatted data.

When training the SVR, the C-value tells how much incorrectly classified values should be punished. This means that a high C-value creates a stricter classification model, but also makes the model creation more time-consuming, and can lead to overfitting. Manual testing in Weka showed that a C-value between 0 and 10 usually gave good results.

Running the SVR classifier is very time consuming, and some shortcuts had to be made. In order to lower the number of needed parameters, the first test that was run simply tested the four kernels on C-values from 0 to 10 (with an increment of 0.1). This test showed that both polynomial kernels were a lot slower than the RBF- and PUK-kernels, and that the RBF-kernel gave the lowest average MAE in this range of C-values, as seen in Figure 3. It was also reported earlier that the RBF-kernel generally gave good bus arrival prediction results in the traffic literature [7]. For these reasons, the RBF-kernel was chosen as the kernel to test more thoroughly.

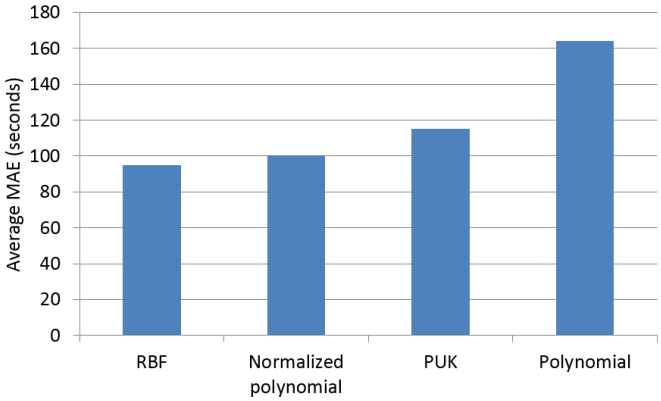

Fig. 3: For each kernel: average MAE with C-values from 0.1 to 10 on the 23 representative datasets.

In addition to the C-value, the RBF-kernel

has a γ-value that also had to be optimized. The γ-value decides how much influence each instance has [25]. To find the best combination of the C-value and the γ-value, five gradually more detailed grid searches were performed. A grid search is a simple search method for searching a large space in a short time, by starting out with a very sparse search, then zoom in on the most promising area, and repeat the process with a more and more detailed search [26].

The first grid was designed to search the entire feasible range of C- and γ-values, but very sparsely. See Appendix C, Table A4 for a detailed overview of the search spaces and step sizes of the five grids that were used.

The ratings for all values of the fifth search had no clear gradients. Because of this, the search was ended, and the best values were selected to be: $C = 5.717, \gamma = 0.0256$.

3.4 Comparison of default and optimized parameters

As seen in Figure 4, the parameter optimizations made the classifiers perform better on most of the groups. kNN improved slightly, while ANN and SVR had major improvements; the overall MAE was lower and much more stable with the improved settings.

It is important to note that the classifiers did not perform better on each and every dataset after this optimization. Since the optimal parameters were chosen by averaging over all representative datasets, the improved settings was better on average, but not necessarily on every single dataset. This is sufficient when the goal is to compare the classifiers against each other on all datasets, but in practice, the parameter optimization should be run on a single dataset type to find the best parameters for that configuration.

The SVR classifier had a particularly high average MAE on all datasets before the parameter optimization, due to the polynomial kernel being the default kernel in Weka. With this kernel, SVR had problems classifying certain sets, resulting in extreme MAE-values that skewed the average.

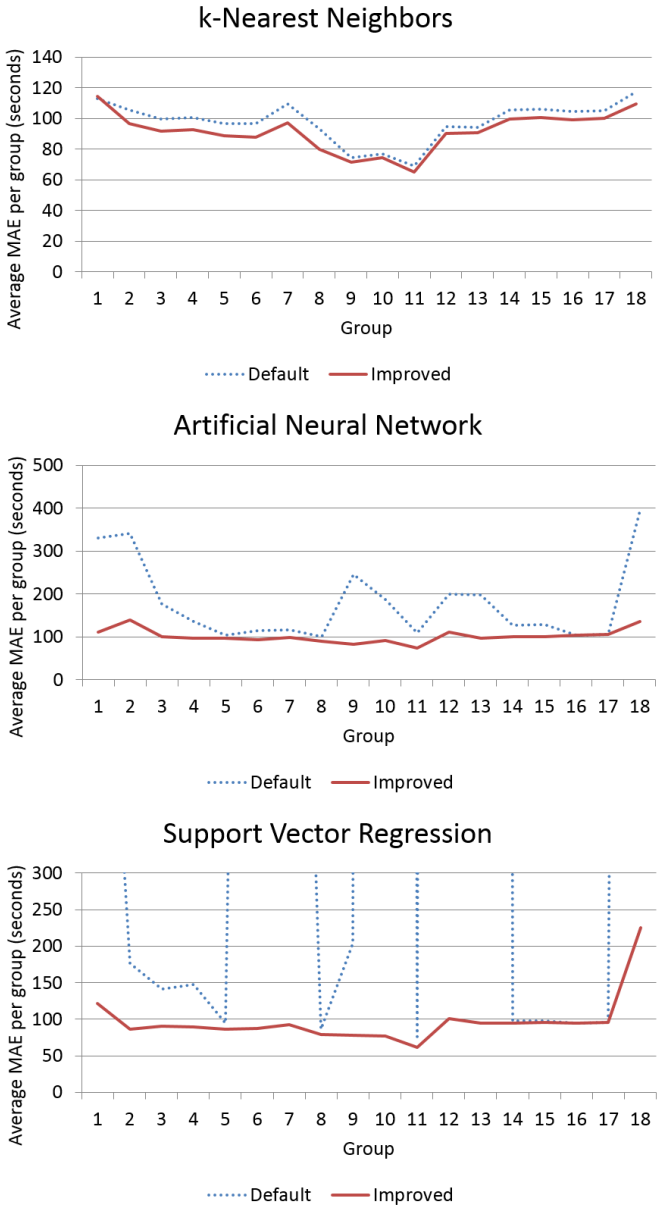

Fig. 4: The default classifier settings versus the optimized settings averaged on all datasets in the 18 groups. Note that the default SVR settings has several huge spikes that disappears outside of the plot.

4 Dataset Analysis

The datasets that were assembled for this work were designed to test a wide range of different possibilities. This included the length of the training and testing period, which was tested from a few hours up to a hundred days. The distance between the training and testing period was tested, and ranged from letting the testing start immediately after the training ended, to testing almost three months later.

Finally, the connection between the training and testing periods was tested with different training systems – some of the training periods were just a single continuous period of time, while others consisted of consecutive recurring days to see if that improved the prediction accuracy.

This chapter will analyze the results from all the different dataset types. To ease the reading experience, a subset of Table 1 containing the currently discussed groups is included in each section.

4.1 Training on a few hours

Table 3: A subset of table 1, listing the dataset types discussed in this section.

ID	Description
1a	Train on one hour, test on the following hour.
1b	Train on two hours, test on the following hour.
1c	Train on three hours, test on the following hour.
1d	Train on four hours, test on the following hour.
1e	Train on five hours, test on the following hour.

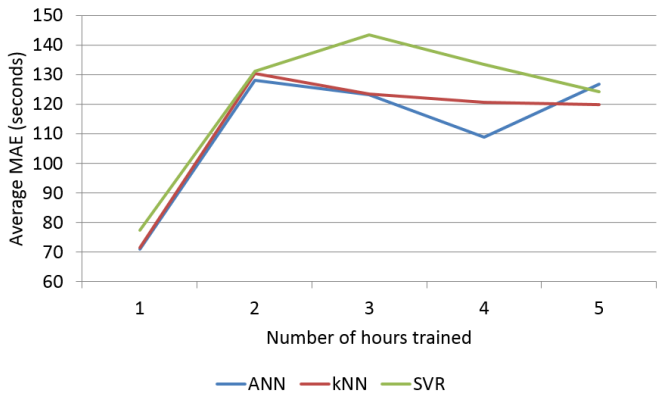

Fig. 5: The three classifiers trained on 1-5 hours, and tested on the following hour.

These datasets were made to test how well the classifiers can perform on small amounts of data. The results must be analyzed with caution, since there is only about 4 bus stop passages per hour. In addition, some of the data sets crossed the two daily traffic peaks (when people travel to and from work), and may have been influenced by this. When trained on longer time periods, the classifiers are assumed to learn to predict the daily peaks on their own by using the TimeOfDay attribute, without being manually configured for it. This is one of the benefits with data-driven approaches. It is worth noting that ANN performed best on these very small sets, with kNN a few seconds behind. The 1 hour sets gave a much lower MAE than the 2-5 hour sets, but since the 1 hour sets only had very few instances, this is not enough to make any rigorous conclusions. The 2-5 hour sets had a potential pattern of decreasing MAE (especially kNN, as seen in Figure 5), which indicates that on such small datasets, the accuracy is improved when the number of training hours is increased. This is probably because training on a too small timespan does not give enough information to create a good model, since only a single corrupted instance will be a large percentage of the dataset.

4.2 Training on a week

Table 4: A subset of table 1, listing the dataset types discussed in this section.

ID	Description
2	Train on Monday-Thursday, test on Friday.
3	Train on Monday-Sunday, test on each of the days the following week.
4	Train on Monday-Friday, test on each of the weekdays the following week.

In group 2, the classifiers were trained on Monday-Thursday in one week, and tested on the Friday in the same week. This group was created to assess the assumption that each day has very different traffic patterns, by training on four different days, and testing on the fifth. With a MAE of 85 seconds, attained by SVR, it can be seen that even though the days were different, the traffic patterns were similar enough to be used for prediction between days.

Groups 3 and 4 were trained on the entire week and the weekdays respectively, and then tested on each of the days the following week. These datasets served two purposes: a) investigating the prediction accuracy when training on an entire week, but testing on a single day, and b) investigating the fact that weekends are traffic-wise very different from the regular

weekdays, and how this may influence the prediction accuracy [27].

Figure 7 shows that the difference in training resulted in a minimal difference in output – kNN trained on five days and kNN trained on seven days performed almost equally when tested on the same days, and the same trend can be seen for the two other classifiers. The classifiers trained on five days performed slightly better, probably because Saturday and Sunday changed the pattern. It is also worth to note that except for kNN on the Monday, all classifiers followed the same pattern, with Monday and Friday being easiest to predict, and Wednesday and Sunday hardest. That weekends are harder to predict is to be expected, since the traffic pattern on these days is very different from the pattern on weekdays.

In the autumn, the normal traffic pattern for cities in Norway is that Monday is the day with the least traffic congestion, and then the congestion increases for each day until the weekend begins [27]. The average delay of week 40 and 47, however, was higher on the Wednesday, which would suggest a high traffic congestion. As seen in Figure 6, which plots the average delay of the buses each day, this pattern was mostly from week 47, while week 40 was almost flat. The higher congestion in the middle of the week may explain the prediction difficulties for these days, while the lower congestion on Mondays and Fridays may explain the good results these days.

SVR and kNN had very similar predictions, and outperformed ANN on these datasets.

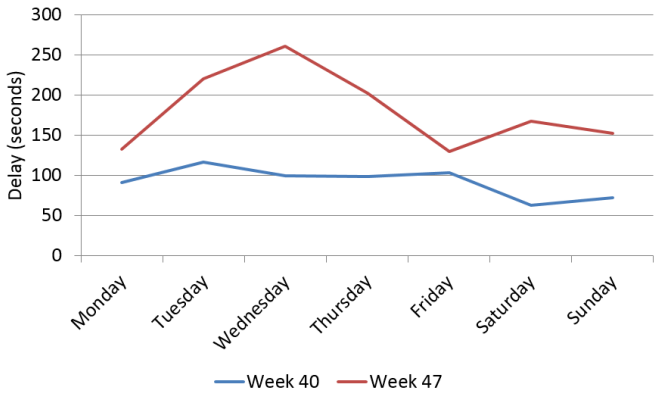

Fig. 6: The average delay of the buses on each day of the test weeks.

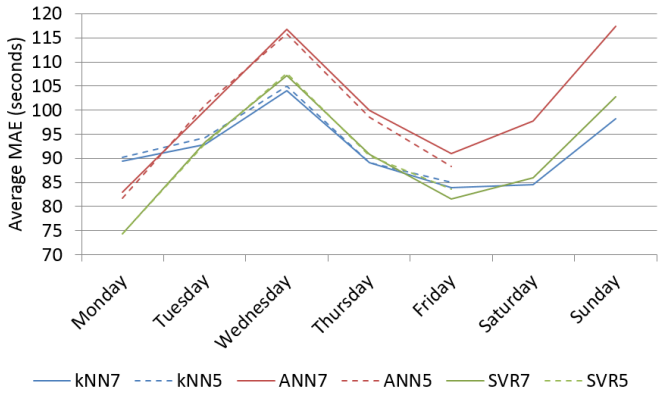

Fig. 7: The three classifiers trained on a full week (x7) and only the weekdays in a week (x5), then tested on each of the days in the following week. Averaged over three different bus stops in two different weeks (weeks 40 and 47).

4.3 Training on multiple weeks

Table 5: A subset of table 1, listing the dataset types discussed in this section.

ID	Description
5	Train on two weeks, test on the following week.
6	Train on three weeks, test on the following week.
7	Train on three weeks, test on the week one week later.
8	Train on three weeks, test on the week two weeks later.

When trained on these datasets, the classifiers had a longer period of time and more data to learn from. This was assumed to improve the prediction accuracy. In addition, the datasets test the assumption that it is easier to predict immediately after the training period ended, than weeks later.

As Figure 8 shows, when testing on the week immediately following the training period, training on two weeks seems to give a lower MAE than training on three weeks, regardless of classifier (the two leftmost bars in each section). This can be explained as overfitting; training on an additional week may cause the classifier models to become unnecessarily complex, resulting in worse predictions.

What cannot be explained as overfitting, though, is the decrease in MAE when expanding the amount of time between the end of the training period and the start of the test period. In Figure 8, the three rightmost bars

in each section represents groups 6, 7 and 8 in the dataset groups respectively. Each of them were trained on the exact same data, but they were tested on the weeks immediately after the training period, and 2 and 3 weeks later respectively. The authors would have expected the MAE to increase when expanding the gap, but the MAE clearly decreased instead. As Appendix D, Figure 21 indicates, this pattern was strongest on the second period (trained on weeks 46-48), while the first period (trained on weeks 36-38) was flatter, but still had a slight decline.

On average, SVR outperformed the other classifiers on these relatively large datasets. The MAE of kNN was usually a few seconds higher, while the MAE of ANN was from 5 to 10 seconds higher.

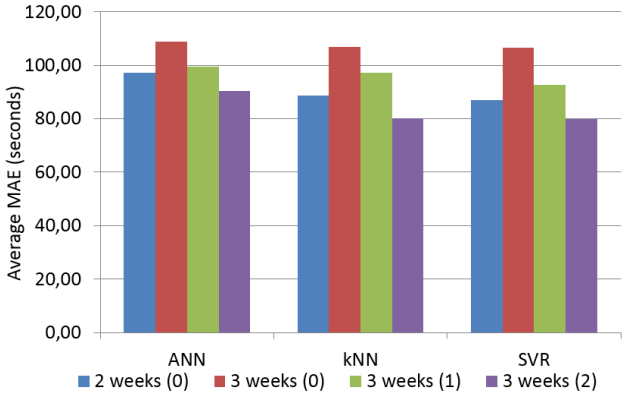

Fig. 8: The three classifiers trained on 2 and 3 weeks, and tested on the week following immediately, 1 and 2 weeks later. *2 weeks (0)* means training on 2 weeks, with 0 weeks between test and training period.

4.4 Training on consecutive recurring days

Table 6: A subset of table 1, listing the dataset types discussed in this section.

ID	Description
9	Train on one day, test on the same day the following week.
10	Train on two consecutive recurring days, test on the same day the following week.
11	Train on three consecutive recurring days, test on the same day the following week.
12	Train on four consecutive recurring days, test on the same day the following week.
13	Train on five consecutive recurring days, test on the same day the following week.

These datasets were based on the fact that the same weekdays show the same traffic patterns [27]. This could mean that it is better to train on Mondays only if the goal is to predict bus arrival times on a Monday. This assumption was assessed by training on 1, 2, 3, 4 and 5 consecutive recurring days, and then testing on the next. For example, one group was trained on five Mondays in a row, then tested on the 6th, while another was trained on five Tuesdays in a row, then tested on the 6th, and so on for the rest of the seven days in the week.

Figure 9 shows the results, here averaged over all days to make a simple plot. The averaged plot shows that there was a general MAE increase when training on 1 to 5 consecutive recurring days, but a sudden decrease when training on three days. When the data behind this plot was analyzed, this trend was visible on all bus stops for all classifiers on all days, even when the classifiers were run on the base attributes only. This could be because the third week was particularly easy to predict, but it could also be an indication that three days is the optimal amount of days to train on in this scenario.

kNN had the best average results on all of these datasets, with an average MAE of 78 seconds. The best result on a single group, however, was 61 seconds by SVR when training on three consecutive recurring days.

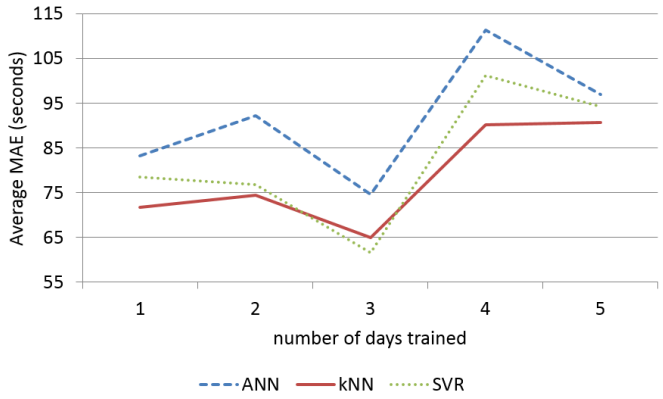

Fig. 9: The three classifiers trained on 1-5 consecutive recurring days and tested on the next, for example trained on 3 Wednesdays and tested on the 4th.

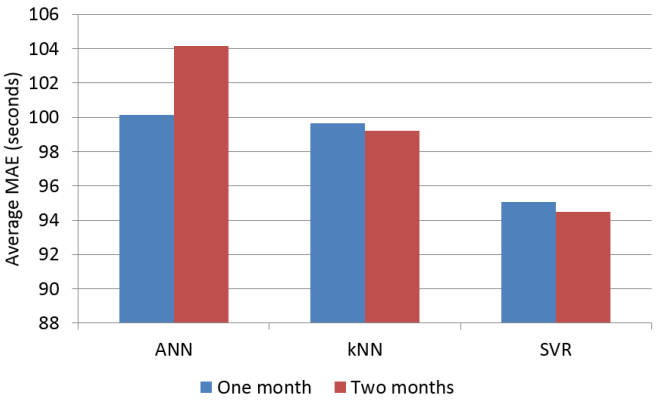

Fig. 10: The three classifiers trained on one or two months, and tested on the immediately following month.

4.5 Training on months

Table 7: A subset of table 1, listing the dataset types discussed in this section.

ID	Description
14	Train on one month, test on the following month.
15	Train on one month, test on each of the weeks in the following month.
16	Train on two succeeding months, test on the following month.
17	Train on two succeeding months, test on each of the weeks in the following month.

These groups were created to test how well classifiers can predict when being trained on relatively long periods, and then tested on either long periods immediately after the training period ended, or shorter periods a while later.

Groups 14 and 16 were trained on one and two months respectively, and then tested on the immediately following month. They were both tested on the same month (November), but group 14 was trained on October only, while group 16 was trained on both September and October.

As presented in Figure 10, the prediction accuracy was very similar regardless of training period. ANN had the largest difference, but even then the MAE difference was less than 5 seconds. This indicates that when trained on a whole month, additional data does not improve the predictions. SVR got the best results, with an average MAE of ca 95 seconds on both groups.

Groups 15 and 17 are trained on the same months as groups 14 and 16, but tested on all four weeks of November instead of the entire month at once. As Figure 11 shows, the difference between training period is again very small, and it is clear that the prediction accuracy decreases for all classifiers as the time gap between testing and training is increased.

Again, SVR attained the best MAE for all datasets in these groups, with a minimum of 73 seconds when tested on the first week of November, regardless of training period. ANN and kNN were not far behind, but the differences were clear. It is interesting to note that both kNN and SVR performed extremely similar when trained on one and two months, suggesting that they used the same neighbors/support vectors in both cases. ANN does not have this benefit.

4.6 Training on 100 days

Table 8: A subset of table 1, listing the dataset types discussed in this section.

ID	Description
18a	Train on 100 succeeding days, test on the 1, 2, 7 and 14 following days.
18b	Train on 100 succeeding days, test on 1, 2, 7 and 14 succeeding days approximately three months later.

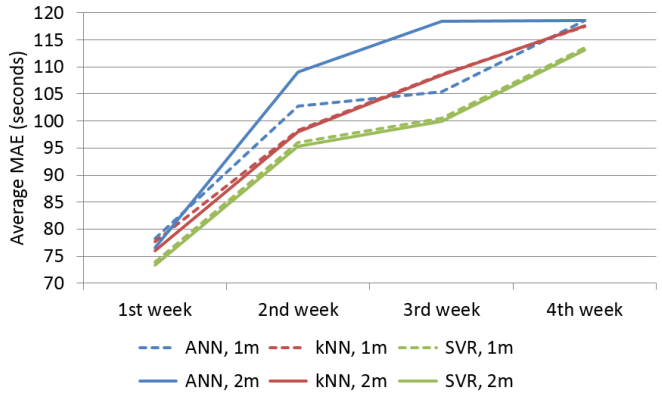

Fig. 11: The three classifiers trained on one or two months (1m and 2m), and tested on the four weeks of the immediately following month.

Fig. 12: The three classifiers trained on 100 successive days, and tested on periods of 1, 2, 7 and 14 successive days either immediately after the training period ended, or 2.5 months later.

All datasets in these groups were trained on the exact same 100 days, going from September 1st to December 8th. These datasets were assembled to investigate how the classifiers perform when trained on a very long time period, possibly going from a summer/autumn traffic pattern, and over to a winter traffic pattern.

The trained classifiers were tested on the 1, 2, 7 and 14 days immediately following the training period, and then on the same amount of days in February 2014. As Figure 12 shows, the MAE decreased when increasing the size of the testing set. One possible explanation is that the first days of the testing set were particularly hard to classify, and that the periods of length 7 and 14 contained easier days that lowered the average.

A more interesting result was that kNN achieved better results when tested on the datasets of length 7 and 14 in 2014, than it did on the corresponding sets from 2013. This can also be explained with hard or easy days, but it is still very interesting that it is possible to get an average MAE of 85 seconds when the testing period is more than two months from the training period. SVR, on the other hand, had great difficulty with the testing sets in 2014, but made the trend with a decreasing MAE when increasing the testing set length very obvious.

4.7 Discussion

Table 9 summarizes the results from the dataset analysis. As illustrated, there was not much variation between the different types of datasets. This is probably because the gap between the time of the last given information to the actual arrival of the bus is small (usually 15-30 minutes, as discussed in Chapter 1). The average delay from all datasets is 144 seconds, and the standard deviation is 184 seconds. This means that with average MAEs between 60 and 80 seconds, the predictions are good, and far from being simple averages.

The results indicate that training on the same kind of day that is to be predicted is reasonable, and that training on three in a row is enough. This makes intuitive sense, since the traffic pattern is very similar on consecutive recurring days, but still changes with different times of the year [27]. Too few training days may not give the classifier enough data to create a durable model, while too many training days may conflict with the periodically changing traffic pattern.

SVR and kNN both performed well on most

Table 9: A summary of the data set analysis.

Section	Training on	Best MAE	Best classifier	Comments
4.1	A few hours	70 seconds	ANN	Training on a single hour gave the lowest MAE, but with only 4 data points, it is difficult to be certain. Training on 2-5 hours gave similar results.
4.2	A week	86 seconds	SVR	SVR and kNN perform very similarly on these sets, but on average, SVR is slightly better.
4.3	Multiple weeks	80 seconds	SVR	kNN only slightly worse than SVR on these sets. Training on two weeks gave a lower MAE than training on three weeks, except when the testing period starts later (strangely enough).
4.4	Days of same type	61 seconds	kNN	All classifiers show a higher MAE when trained on a higher amount of days, except when trained on three days, which gives the minimum.
4.5	Months	73 seconds	SVR	Best performance when trained on 2 months and tested on the first week of the 3rd month. Very small difference between being trained on 1 and 2 months.
4.6	100 days	74 seconds	kNN	The best results were attained when tested on 2 weeks. kNN best when gap between training and testing, SVR best without gap.

of the dataset types, while ANN usually attained a higher MAE. It is also worth noting that on the groups that kNN did not have the lowest MAE, it was usually very close. This indicates that kNN performs well on data from the traffic domain. Both SVR and kNN generally gave good results, suggesting that the use of neighbors/support vectors helped them find the most important cases, while ANN struggled to create good models. Future studies should investigate the most interesting dataset types in the dataset analysis further, as this work merely scratches the surface.

In addition, kNN was clearly the fastest classifier. SVR was about five times slower than kNN when both methods employed optimized parameters, and ANN was almost a 100 times slower when using the $49, 49, 49$ network. However, if speed is more important than accuracy, using $3, 3, 3$ as the network topology for ANN achieved a running time slightly lower than SVR with approximately the same rating as $49, 49, 49$.

kNN was also the classifier that gave the best results before the parameter optimization, without any MAE spikes. After the optimization, the MAE was very similar, but slightly lower. This proves that kNN is not as dependent on the perfectly optimized parameters as the other classifiers. Thus, kNN proves to be both precise, fast, and robust.

All classifiers were lightning fast when testing – even in the extreme cases where the training took several weeks (some SVR configurations on the largest datasets), the testing was still done in less than a second.

5 Attribute analysis

The purpose of an attribute analysis is to assess the level of importance for the attributes used in the datasets. The analysis was done by comparing the prediction MAE of only the base attributes with the MAE of the base attributes combined with additional attributes for weather, football, ticket, and passenger data. The base attributes and the additional attribute sets are listed in Appendix B, Table A1.

The inclusion of the attributes TicketValidationsLastThreeHours and FootballMatch always resulted in reduced, or at best unchanged accuracy, and was therefore ignored in the analysis. As for the football matches, the increased congestion close to Lerkendal stadium did not influence the bus delays noteworthy. This may have been due to the separation of public and private transport lanes in the area. Note that these two attributes still are a part of the all attributes set, since they were included in the initial processing.

Since this analysis was run in parallel with the parameter optimization, the optimal pa-

rameters were not yet discovered. Therefore, all classifiers were trained with the default parameter settings. The following sections will explain important trends and observations over the different datasets for the different classifiers.

5.1 k-Nearest Neighbor attributes

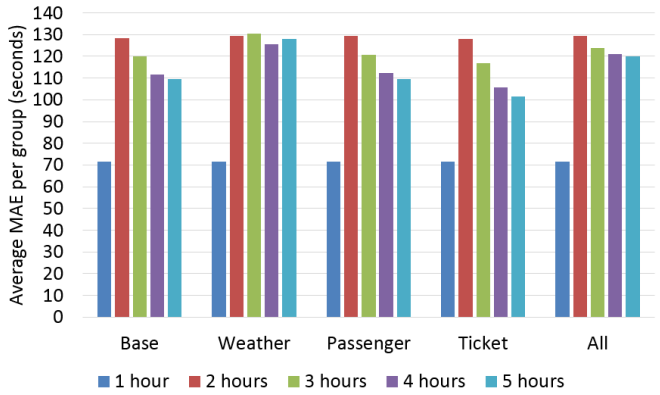

Fig. 13: Training kNN on depicted number of hours. Tested on the following hour.

When training on a few hours, the limited amount of training instances caused the predictions to fluctuate. Still, as Figure 13 shows, a pattern of declining MAE could be observed. Note that ticket data did not change the outcome of the predictions for 1 and 2 hours, but increasingly improved the predictions for 3 to 5 hours. Clearly, the weather did not improve the predictions.

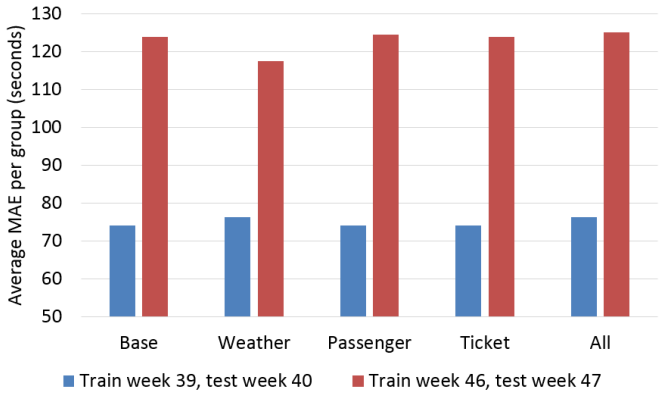

Fig. 14: Average Attributes MAE for dataset group nr. 3 (Train on Monday-Sunday, test on each of the days the following week) with the kNN classifier.

In group 3, training was done on Monday-Sunday and tested on each of the days in the following week. This was done in two time periods; weeks 46/47, and weeks 39/40. When training on week 46 and testing on week 47, the MAE of the base attributes was 67 percent higher than when training on week 39 and testing on week 40. Why week 47 was harder to predict can be explained with the abrupt temperature change in the middle of week 47, which may have posed a challenge for how kNN picks the relevant neighbors.

Note that the predictions of kNN were slightly improved when adding the weather attributes in week 47. A possible explanation can be found in the temperature change, where temperature dropped to minus degrees centigrade. Adding the weather attributes reduced the average MAE slightly for week 47, but increased the MAE for week 40 where the temperature was stable. See Appendix E, Figure 22 for a detailed temperature overview. Note that each of the weather attributes separately did not reduce the MAE as much as all combined.

5.2 Artificial Neural Network attributes

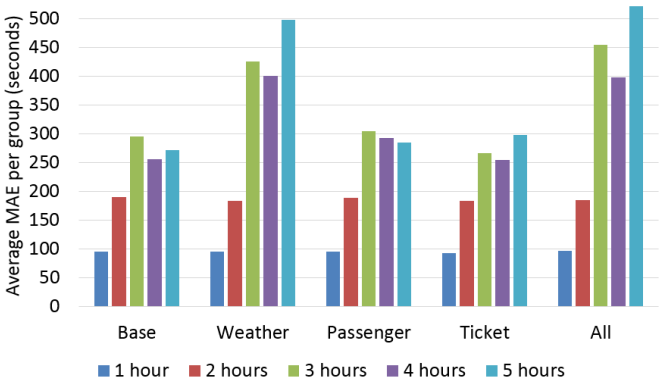

Fig. 15: Training ANN for hours.

As Figure 15 shows, the MAE of the ANN classifier was notably increased when weather attributes were included for hourly training. The same can be seen to a lesser degree when the training was done on a set of consecutive recurring days, for example training on a set of Mondays, and testing on the last, as shown in Figure 17.

For training on multiple weeks, adding the passenger and ticket attributes made ANN perform slightly better when the test week

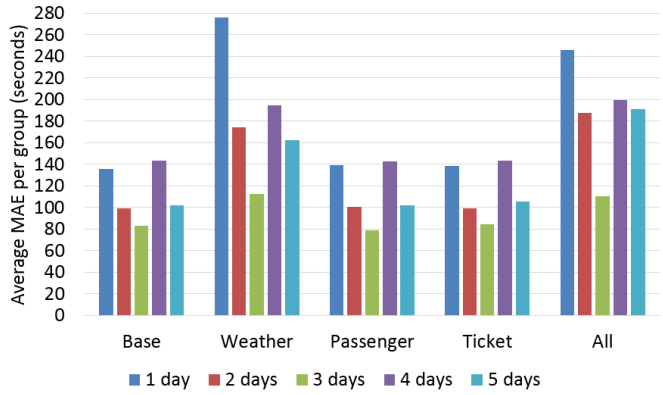

Fig. 16: Training ANN on 1-5 consecutive recurring days and tested on the next, for example trained on 3 Wednesdays and tested on the 4th.

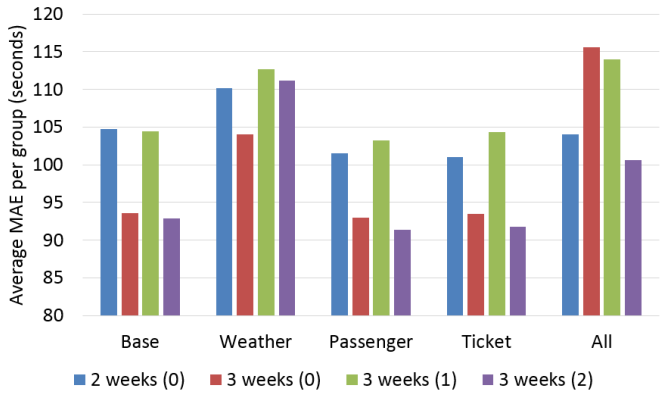

Fig. 17: Training ANN on depicted number of weeks. *2 weeks (0)* means training on 2 weeks, with 0 weeks between training and test period.

immediately followed the training weeks, as presented in Figure 17.

5.3 Support Vector Regression attributes

Adding extra attributes did not seem to noteworthy change the prediction performance for SVR, as seen in Appendix F. There were some minor differences, but not enough to draw any conclusions. This may be a result of how SVR uses support vectors to classify future instances. If the additional attributes does not improve the model, SVR will use mostly the same (or similar) instances as support vectors, and the classifier will use a similar hyperplane for classification [7].

5.4 Discussion

There were very few large football matches in the training period, since this is a relatively small city. The passenger counting was also poor, since only one bus had it installed, and since it had not been tested long enough to actually be known to work at the time of writing. As such, more studies should be performed to investigate the actual influence of such attributes.

When training on 3-5 hours, the ticket attributes improved the prediction accuracy of kNN. In contrast, the inclusion of weather attributes reduced the accuracy. The overall MAE for the base attributes decreased as the number of hours (and thereby the number of instances in the datasets) increased. This makes sense, as the relative weighting of each neighbor is smaller, reducing the importance of outliers.

Regardless of training period, it is clear that the ANN classifier does not perform better when using weather. When training on hours and consecutive recurring days, weather reduces the prediction accuracy. A reason for this may be that with hourly training, the weather data is superfluous and the classifier fails to generalize on the smallest datasets, while on the longer, the weather data is so different that the classifier is unable to see a connection between the weather and the delays.

The SVM classifier is stable and performs well on all attribute sets. This may be a result of the classifier's ability to ignore attributes not improving the model. There are minor differences, but these can not be used to draw any conclusions.

6 Real-Time Prediction

In Chapters 4 and 5, the predictions on the different datasets and attributes rarely resulted in mean errors below 70 seconds. As discussed in Chapter 1, these predictions are typically given 15 to 30 minutes prior to the bus arrival. However, by utilizing real-time information such as when the bus passes different bus stops, the predictions can be improved as the bus progresses through its route. Due to the frequent bus stop passages in urban areas, this

approach provides near real-time predictions, benefiting waiting passengers.

Late findings, as illustrated in Appendix G, indicated that the correlation between travel times of consecutive trips was higher than between delays of consecutive trips. In addition, travel times accumulate at passages after departure, starting from zero, while a delay may have accumulated prior to departure, and does not necessarily increase during the trip. For these reasons, the prediction error was hypothesized to decrease when predicting travel times instead of delays. The POC model assessed this hypothesis by predicting travel time to a stop once the bus had started moving, and then at all intermediate bus stops. The model was conceptual, in the sense that it was provided with simulated data from an interface and infrastructure that were both hypothetical.

The POC model utilized a classifier (kNN, ANN or SVM) for its predictions. Hence it needed to do training and testing on instance-based data where every instance represented a single trip. The following set of attributes were utilized from the trips:

- Travel times to consecutive bus stops (accumulated by arrival times, in seconds). The last stop and consequently the total travel time was the class attribute.
- Stopping time at bus stops (accumulated to the respective stop, in seconds)
- Time of departure (in hours)
- Scheduled travel time (derived from scheduled arrival time, in seconds)
- Average travel time for the three previous buses

The instances were split into a single training and test set for every stop between the departure stop and the stop for which the travel time was to be predicted. Hence, it was possible to simulate real-time by predicting the total travel time for every bus stop passage.

In order to evaluate the POC model, it was compared to the existing real-time system from AtB. The AtB system and the dataset (testing environment) is described in more detail in the following sections, together with a comparison of candidate models for the POC.

6.1 Existing Real-Time System

The existing AtB system at the time of writing includes passenger information displays installed in the most popular stops in the Trondheim region. Every minute, new predictions are conveyed to the displays, informing waiting passengers about line-specific arrival times. In addition, the real-time service is available via AtB's website. Real-time data from this website was fetched in order to compare the models in a simulated real-time environment.

AtB's real-time system is developed by Swarco, who has delivered similar solutions in Spain, Italy, Greece and Sweden [28]. Their model applied for prediction (from now on referred to as the AtB model) should therefore be a worthy competitor for the POC model. The AtB model employs GPS and odometer data for its predictions [29]. However, the details of the implementation remain hidden, and the model is therefore considered a "black box".

6.2 Dataset

In order to take advantage of the findings from the parameter optimization and attribute analysis in Chapters 3 and 5, the same route was selected for prediction. As mentioned in Chapter 2, the many intersections, crossings and stops resulted in a stochastic environment where travel time was hard to predict. Consequently, the predictions became more easily distinguishable when compared to another.

Due to frequent loss of both passage data and AtB predictions towards the end of the route, the three last stops were excluded, and travel times were predicted for stop 17 (*Risvollan Senter*). In addition, the period of testing was set from Monday 17th to Friday 21th of February, 2014. The weekend was excluded due to missing AtB predictions. The resulting test set included a total of 340 trips, or instances.

6.3 Model Selection

When the dataset and attributes were selected, the selection of a prediction model remained. This included the selection of which method to employ (ANN, kNN or SVR), the parameters for the method (topology, kernel values,

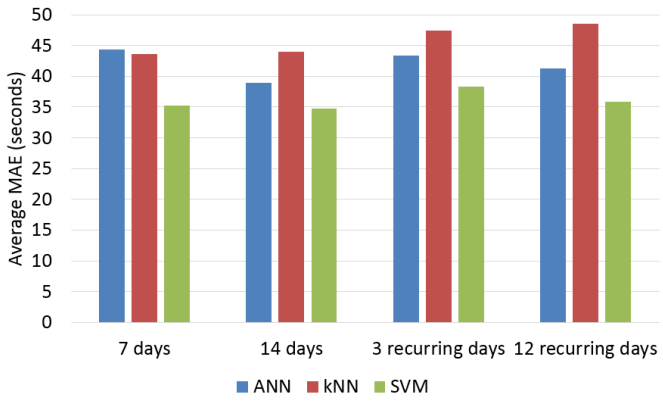

Fig. 18: Average prediction errors on travel time to stop 17, *Risvollan Senter*, at bus stop passages 1 to 16.

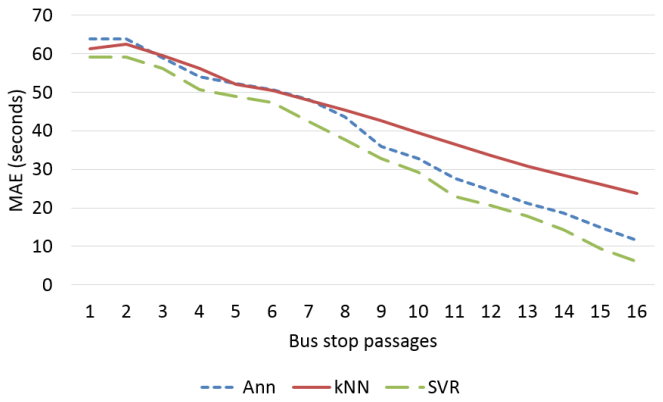

Fig. 19: Prediction errors on travel time to stop 17, *Risvollan Senter*

etc.) and on which days preceding the days of testing to train the method.

The fine-tuning of classifier parameters was avoided by reusing the best parameter configurations for each method found in Chapter 3. Although these parameters were specialized for prediction of delays on other datasets, they were general in the perspective of domain and prediction type.

There were four generated training sets; training on 7 and 14 consecutive days, and training on 3 and 12 consecutive recurring weekdays. The ANN, kNN and SVR classifiers were trained on each of the four training sets, and the methods were compared by predicting the travel times to stop 17, at the time of departure from Kongens Gate K1 and the passages of the 15 intermediate stops. The prediction MAE at each of the 16 stops were then averaged for all predictions on the 5 weekdays of testing.

Figure 18 illustrates the average prediction error for all stops, when each method was trained on the different training sets. The best configurations (ANN and SVR on 14 days, kNN on 7 days) were compared stop by stop, as seen in Figure 19. The results indicated that SVR was the preferred candidate method, with the lowest MAE at all stops. Hence, this SVR configuration was employed in the POC model.

6.4 Performance Comparison

The AtB model predicts about every minute, in contrast to the POC model which predicts at every bus stop passage. In order to compare the models without reducing the accuracy of the AtB model, a simple rule was attached to the POC model: if more than one minute had passed since the last bus stop passage at the time of a new AtB prediction, the POC model would predict the same arrival time as at the previous passage.

Once every now and then, the bus signals are lost. Consequently, if the signal for the bus closest to arrival is lost, the AtB model may predict arrival for the next bus, if there is one. Typically the signal is back a minute later. To compensate, every time the AtB model predicted with an error exceeding 5 minutes of the previous prediction (indicating a signal loss), the prediction value was set to the previous prediction, thereby reducing the error.

Figure 20 shows the performance of the POC model and the AtB model, predicting the arrival time at *Risvollan Senter* for the 340 trips on Monday to Friday (17.02.2014-21.02.2014), x minutes after departure from *Kongens Gate K2*. The performances for the individual days are listed in Appendix H. The whiskers are minimum and maximum errors, and the boxplots are first, second (median) and third quartiles. The comparison is illustrated by box- and whisker-plots to facilitate the importance of the error distribution from a travelers perspective; a lower MAE is not necessarily better, if the maximum errors are much higher.

The results indicated that the POC model predicted very similar to the AtB model in the first minutes from departure, with a slightly lower MAE, but with higher maximum errors. However, in the minutes closer to arrival,

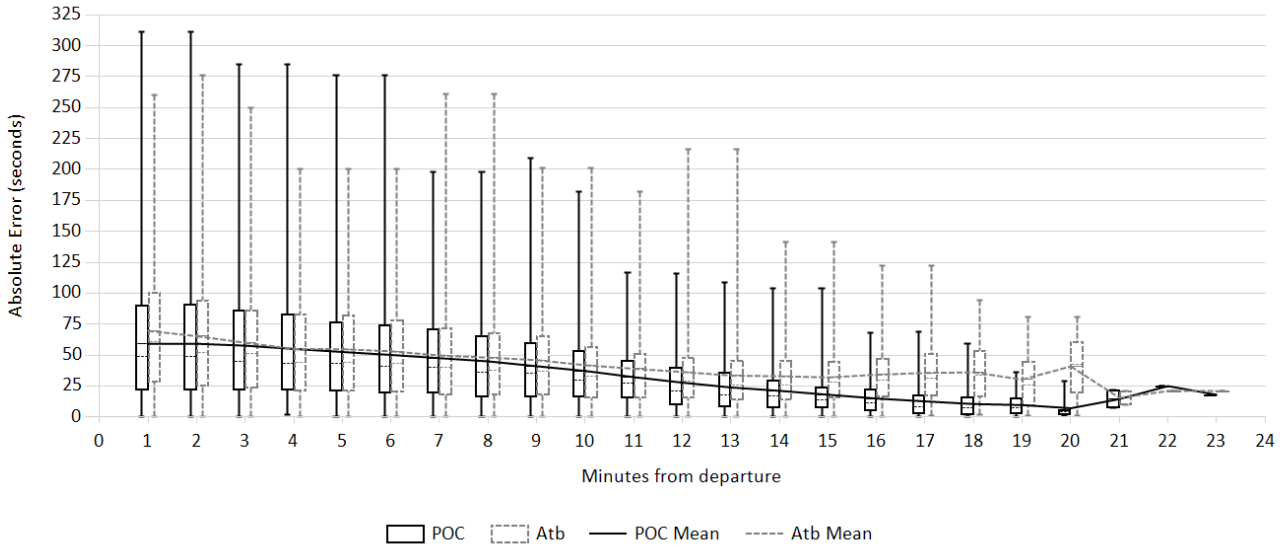

Fig. 20: Prediction errors on travel times to *Risvollan Senter* on trips from *Kongens Gate K2*, 17.02.2014-21.02.2014

the POC model clearly outperformed the AtB model with both lower variance and MAE. As a bus progressed on its route, the travel time profile of the respective instance became more detailed (travel times and stopping times for all stops passed on the trip were included). As a result, the prediction accuracy increased closer to arrival. This was a clear tendency on all days in the testing period. The average travel time for the period was approximately 17 minutes. Consequently, the errors depicted for the last minutes in the figure represented only a few trips. As in this case, the errors may have increased due to the rare occurrence of similar trips in the training set.

6.5 Discussion

Despite the promising performance of the proof of concept model, incorporating the model into a real system would be hard, or even unfeasible, due to limitations in the infrastructure. A real system would require an extension of the model to predict travel times to all consecutive stops on a route. In the case of the POC model, a straightforward extension would require 16 predictions to stop 17, 15 predictions to stop 16, and so on. In general, a route with n stops would require $\sum_{n=1}^{n-1} n$ classifiers. Consequently, a route with 30 stops require $\sum_{n=1}^{30-1} n = 435$ prediction models. Although training this amount of classifiers may be feasible at say, every night to Monday, the system would become unnecessary complex with respect to data logistics and scalability. Hence, an alternative approach should be employed in case of production. Also, since predictions are done per passage, the model may not perform equally well in suburban areas, as the distances between stops typically increase.

7 Conclusion

The parameter optimization improved the prediction accuracy of all classifiers, particularly ANN and SVR, which had multiple occurrences of very high MAE before the optimization. The MAE was smoothed and lowered after optimizing the parameters. This shows that optimizing the parameters is an important part of preparing for arrival time prediction, especially for ANN and SVM.

While all classifiers performed well, kNN and SVR attained the best predictions on most of the datasets. Furthermore, kNN clearly had the shortest training time and its predictions was more stable with respect to parameter tuning. Hence, it is suitable for real-time demanding and constantly changing traffic environments.

The dataset analysis found that the best way to predict the arrival time on one particular

day, is to train on the three previous days of the same type. This means that if the goal is to predict for the coming Wednesday, the classifier should be trained on the previous three Wednesdays.

Adding weather information gave no clear prediction improvements. The amount of people taking the bus each day is so large that even when some of them decide not to take the bus, there could very well be an equally large group that would otherwise walk or cycle taking it instead. Passenger and ticket data has promising results for some of the training periods for ANN and kNN.

It was indicated by the proof of concept model that use of travel time rather than delay, improves prediction accuracy. Furthermore, the proof of concept model was able to predict travel time with an accuracy and stability competitive to the existing real-time system by employing SVR and bus stop passage data. In addition, the model is openly explained and the source code is freely available, which is beneficial for further study and improvements [30].

8 Future work

The following future work is suggested:

1) Further investigating the most promising dataset types. Any training sets with duration of a few hours should be planned in such a way that they avoid the large pattern differences from the rush hours to the regular hours.
2) Investigating how to weight the different factors in the rating function to make the average user trust such a system.
3) Running a parameter optimization on kNN with only the best datasets, to further improve the prediction accuracy.
4) Datasets for passenger counting, ticket data and football matches were sparse. Investigate if better data would improve predictions.
5) No clear improvements with additional attributes for SVM was found. This should be analyzed further
6) Investigate how weather affects passengers' travel plans and how this may influence bus arrival times.
7) In order to scale the model to predict travel times to multiple bus stops, alternatives that reduce the number of classifiers required should be investigated. This includes multiple outputs per classifier, interpolation of predictions, and partly substitution of classifers by rules based on domain-specific knowledge.

Acknowledgments

The authors would like to thank Jo Skjermo for being an exceptional mentor, and Tomas Levin for contributing with his domain knowledge. We would also like to thank AtB and NPRA for providing us with large amounts of data, and being very helpful when we needed clarifications. The processing power was funded by the SMIO-project [31].

References

[1] Intelligent transport systems handbook. [Online]. Available: http://road-network-operations.piarc.org/index.php?option=com_content&task=view&id=38&Itemid=71&lang=en [Accessed: 2014-05-09]

[2] K. Manolis and D. Kwstis, "Intelligent transportation systems - travelers' information systems the case of a medium size city," in *Mechatronics, 2004. ICM '04. Proceedings of the IEEE International Conference on*, 2004, pp. 200–204.

[3] Y. Zhengxiang, X. Guimin, and W. Jinwen, "Transport volume forecast based on grnn network," in *Future Computer and Communication (ICFCC), 2010 2nd International Conference on*, vol. 3, 2010, pp. V3–629–V3–632.

[4] C.-H. Wu, C.-C. Wei, D.-C. Su, M.-H. Chang, and J.-M. Ho, "Travel time prediction with support vector regression," in *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, vol. 2, 2003, pp. 1438–1442 vol.2.

[5] R. Jeong and L. Rilett, "Bus arrival time prediction using artificial neural network model," in *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, 2004, pp. 988–993.

[6] A. Nurhadiyatna, B. Hardjono, A. Wibisono, W. Jatmiko, and P. Mursanto, "Its information source: Vehicle speed measurement using camera as sensor," in *Advanced Computer Science and Information Systems (ICACSIS), 2012 International Conference on*, Dec 2012, pp. 179–184.

[7] J. Skjermo, T. Levin, A. Sjåfjell, E. Dahl, and S. Skogen, "Artificial intelligence in bus travel time prediction – a state of the art review," *Under submission to Transportation Research Board (TRB) Annual Meeting 2015.*, 2014.

[8] N. B. C. Norwegian Meterological Institute. Information about yr.no. [Online]. Available: http://om.yr.no/about/ [Accessed: 2014-05-19]

[9] N. P. R. Administration. Information on travel times. [Online]. Available: http://www.reisetider.no/reisetid/static.html?method=english [Accessed: 2014-05-25]

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explorations*, 2009. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/

[11] Amazon. Amazon ec2. [Online]. Available: https://aws.amazon.com/ec2/ [Accessed: 2014-05-12]

[12] L. Vanajakshi and L. Rilett, "A comparison of the performance of artificial neural networks and support vector machines for the prediction of traffic speed," in *Intelligent Vehicles Symposium, 2004 IEEE*, 2004, pp. 194–199.

[13] D. Zheng and Y. Wang, "Application of an artificial neural network on railway passenger flow prediction," in *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, vol. 1, 2011, pp. 149–152.

[14] D. Aha and D. Kibler, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[15] J. K. Uhlmann, "Satisfying general proximity/similarity queries with metric trees," *Information Processing Letters*, vol. 40, no. 4, pp. 175–179, November 1991.

[16] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *ICML'06: Proceedings of the 23rd international conference on Machine learning*. New York, NY, USA: ACM Press, 2006, pp. 97–104.

[17] A. Moore, "A tutorial on kd-trees," Extract from PhD Thesis, Tech. Rep., 1991.

[18] Z. Boger and H. Guterman, "Knowledge extraction from artificial neural network models," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 4, Oct 1997, pp. 3030–3035 vol.4.

[19] K. L. Priddy and P. E. Keller, *Artificial Neural Networks: An Introduction*, illustrated ed. SPIE Publications, 2005, p. 43.

[20] A. Blum, *Neural Networks in C++: An Object-Oriented Framework for Building Connectionist Systems*, 1st ed. Wiley, 1992, p. 60.

[21] S. Karsoliya, "Approximating number of hidden layer neurons in multiple hidden layer bpnn architecture," 2012.

[22] Wikibooks. Artificial neural networks/neural network basics. [Online]. Available: https://en.wikibooks.org/wiki/Artificial_Neural_Networks/Neural_Network_Basics#Learning_Rate [Accessed: 2014-04-23]

[23] C. E. Corporation. Neural network technology. [Online]. Available: http://www.cheshireeng.com/Neuralyst/nnbg.htm [Accessed: 2014-04-23]

[24] J. M. D. Rumelhart, *Parallel Distributed Processing:Explorations in the Microstructure of Cognition: Foundations*. MIT Press, 1987, p. 330.

[25] Machine learning in python. [Online]. Available: http://scikit-learn.org/stable/modules/svm.html#kernel-functions [Accessed: 2014-05-21]

[26] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," *Department of Computer Science, National Taiwan University*, 2003.

[27] "Håndbok om trafikkberegninger," Dec. 1988.

[28] Swarco. Public transport references. [Online]. Available: http://www.swarco.com/en/References/PUBLIC-TRANSPORT [Accessed: 2014-05-16]

[29] U. K. Thorenfeldt, D. Bertelsen, and L. Øvstedal, "Evaluering av enkelte kollektivtrafikktiltak i trondheim vinteren 2011," Aug. 2006.

[30] A. Sjåfjell, E. Dahl, and S. Skogen. On implementations of bus travel time prediction utilizing methods in artificial intelligence - source code. [Online]. Available: https://bitbucket.org/asjafjell/mastergutta/ [Accessed: 2014-06-04]

[31] SINTEF. The smio-project. [Online]. Available: http://www.sintef.no/SINTEF-Teknologi-og-samfunn/Prosjekter-i-SINTEF-Teknologi-og-samfunn/Prosjekter-SINTEF-TS-2013/Smidig-mobilitet-i-Oslo/ [Accessed: 2014-01-01]

APPENDIX A
DATASET AGGREGATION DETAILS

The final 423 datasets used for testing were all created from a single, aggregated data table that combined all the data sources.

Each row in this table represents a row from an AtB table that contained a log of all passages at the different bus stops. This table was filtered so that it contained only passages where a bus on line 8 passed one of the three selected bus stops.

For each remaining row, the following data was added (see Appendix B, Table A1 for an overview of all attributes):

1) **Football match data:** all data rows that were within 2.5 hours from the middle of the football match got the number of spectators of this match as the value of the FootballMatch attribute. All other rows got a 0 on this attribute.
2) **Weather data:** all data rows got the values of Temperature, Wind, Humidity and Precipitation from yr.no as values of the corresponding weather attributes. These values were all collected on an hourly basis (a limitation of yr.no), so multiple data rows may have the same values if they are close enough in time.
3) **Passenger counting:** all data rows that were from the bus with a passenger counting system got the current number of passengers as the value of the PassengersPresent attribute. All other rows got a 0 on this attribute.
4) **Ticket data:** all data rows that were within the two weeks were ticket data was provided by AtB got the number of tickets validated the last 1 and 3 hours as values of the corresponding ticket attributes. All other rows got a 0 on these attributes.
5) **Delay aggregations:** all data rows got multiple aggregations of previous delays at the current bus stop. These values were calculated by picking previous data rows by some criteria (for example the last 3 for AverageLast3, and the last three trips on the same type of day for AverageLast3OfSameDay) and averaging them. Since ScheduledArrivalTime was used to identify data rows, previous rows were selected by this attribute.

APPENDIX B
DATASET ATTRIBUTES

Table A1: The exact attributes of the datasets.

Attribute	Comment	Reasoning	Attribute type
DayOfWeek	The day of week for this bus trip.	The traffic follows approximately the same pattern on the same weekdays.	Base
TimeOfDay	The scheduled arrival time for this bus, measured in quarters since midnight.	Same as above.	Base
StopTime	The duration of which the previous bus stopped on this bus stop. Measured in seconds.	Knowing the previous stop time at the bus stop gives an indication of how much stop time the next bus will need.	Base
Delay	Delay of the actual arrival of the bus. Measured in seconds. This is the class attribute.	The class attribute. We want to predict how delayed the bus going to be.	Base
DelayLast	Delay of the previous bus was at this bus stop.	If preivous bus that arrived at this stop was delayed, this one may be so too.	Base
AverageDelayLast3	The average delay of the previous 3 buses that arrived at this bus stop.	Same as above.	Base
DelayYesterday	The delay of the bus with this ScheduledArrival time from yesterday (or the previous day it was driven).	If bus(es) at this time of day tend to be delayed, this one may be so too.	Base
DelayDayBeforeYesterday	The delay of the bus with this ScheduledArrival time from the day before yesterday (or the previous day it was driven).	Same as above.	Base
DelayLastWeek	The delay of the bus with this ScheduledArrival time from exactly one week ago (or the previous week it was driven).	Same as above.	Base
AverageDelayLast3OfSameDay	The average delay of the buses with this ScheduledArrival time from the last three weeks.	Same as above.	Base
PassengersPresent	The number of passengers present at the previous bus stop.	The number of passengers present will influence the stop time, since more passengers need to get on or off.	Passenger
Temperature	The temperature of the hour of ScheduledArrival. Measured in degrees celsius.	Influences the amount of people that prefer to travel by bus.	Weather
Precipitation	The precipitation of the hour of ScheduledArrival. Measured in millimeters.	Same as above.	Weather
Wind	The wind of the hour of ScheduledArrival. Measured in meters per second.	Same as above.	Weather
Humidity	The humidity of the hour of ScheduledArrival. Measured as a percentage.	Same as above.	Weather
FootballMatch	The number of spectators if there was a soccer match within 2 hours from ScheduledArrival. Otherwise zero.	A high amount of people driving cars and/or buses to the soccer stadium will influence the traffic around it.	Football
TicketValidationsLastHour	The number of ticket validations the last hour.	If a lot of tickets were validated, we can expect more traffic and higher delays.	Ticket
TicketValidationsLastThreeHours	The number of ticket validations the last three hours.	Same as above.	Ticket

APPENDIX C
PARAMETER OPTIMIZATION DETAILS

C.1 k-Nearest Neighbor

The k-value can be anything from 1 to the number of instances in the training set. If it is equal to or larger than the number of instances, all instances are used, which effectively reduces the kNN to a simple averaging function.

The k-values selected for testing were static k-values from 1 to 100, in addition to dynamic k-values that were based on n, which is the number of instances in the training set. The dynamic k-values ranged from $n/500$ to n, with steps of 10 on most of the range (more detailed closer to n). The k-value was rounded to the closest integer, and increased to 5 if it was lower than that (as our tests showed that it was always preferable to use more than one neighbor).

C.2 Artificial Neural Network

Table A2: The network topology of the test networks. The first cells of each row list the topology, with eacy layer separated with a comma. In cases where the amount of nodes was a decimal number, it was rounded downwards.

Topology	Values
n	n from 1 to 25, and 30, 50, 100, and a
5,n	n from 1 to 10
10,n	n from 1 to 10
a,n	n from 1 to 10, and 20, 30, 50 and 100
n,10	n from 20 to 100, with an increment of 10
n, n, n	n from 1 to 100
$n, \frac{n}{2}, \frac{n}{5}$	n from 10 to 100, with an increment of 5

The following networks were tested:

- One hidden layer with 1 to 100 nodes. The dataset dependent value a (attributes + classes /2) was also tested.
- Two hidden layers, the first with a nodes, the second with different numbers of nodes.
- Three symmetric hidden layers with n nodes in each, where n went from 1 to 50.

The networks with one single hidden layer got a lower rating when increasing the number of nodes to about 30-50. For the larger networks a tendency of better rating for 50 or more nodes in three hidden layers were clear. Therefore, the search was expanded with the following networks:

- Three symmetric hidden layers with n nodes in each, where n went from 50 to 100.
- Three hidden layers with n, $n/2$ and $n/5$ nodes respectively.

Networks with three layers with n, $n/2$ and $n/5$ nodes respectively were added to the search since some networks that had fewer nodes in the second layer did well in the previous search. An assumption was made, that a network does not need as many nodes in the latter layers, since it processes the data more and more for each layer, and wanted to test this.

This search showed that the networks with uniform layers continued to give good predictions until they reached approximately 80 nodes, and then the performance decreased. The search was thus not continued further. The networks with gradually smaller layers did contribute with two of the best performing networks ($75, 37, 15$ and $10, 5, 2$). A search past 100 nodes for the gradually smaller layers were not performed because a decrease in rating from 75 up to 100 nodes was clear.

The last parameter to test for is number of epochs. The optimal value for epochs was found for all of the aforementioned networks with the the values found for learning rate and momentum. Tests with epochs from 1 to 1150 was performed. The Weka default value is 500. For the networks $3, 3, 3$ and $10, 5, 2$ it was clear that we needed to expand to be sure that the best epochs value was chosen. For these two networks we tested up to 5000.

C.3 Support Vector Regression

Table A4 lists the five grids that were used for the parameter optimizing grid search for SVM. Each cell consist of the formula (2^n or n), the range for n, and how much n was incremented for each step within this range ("inc").

Table A3: The five best network topologies, their best rating and the momentum and learning rate that caused this rating, and finally the average training time over all datasets.

Network	**Best rating**	**Learning rate**	**Momentum**	**Epochs**	**Avg. training time**
$49, 49, 49$	151.40	0.1	0.2	250	121 287 ms
$75, 37, 15$	153.41	0.1	0.0	750	68 939 ms
$3, 3, 3$	154.68	0.1	0.7	450	2 147 ms
$57, 57, 57$	156.03	0.0	0.0	200	124 816 ms
$10, 5, 2$	158.46	0.1	0.7	650	5 204 ms

Table A4: The five grid searches performed to find the best combination of C- and γ-values for the RBF-kernel. The step sizes/increments are marked ″inc″ in each cell.

C	γ
$2^n, n \in [-15, 15]$, inc 1	$2^n, n \in [-15, 15]$, inc 1
$2^n, n \in [-1, 4]$, inc 0.2	$2^n, n \in [-6, -3]$, inc 0.2
$2^n, n \in [2, 3]$, inc 0.05	$2^n, n \in [-5.6, 5]$, inc 0.05
$n, n \in [5.657, 6.057]$, inc 0.05	$n, n \in [0.0237, 0.0257]$, inc 0.0005
$n, n \in [5.657, 5.757]$, inc 0.01	$n, n \in [0.0247, 0.0257]$, inc 0.0001

APPENDIX D
DATASET ANALYSIS DETAILS

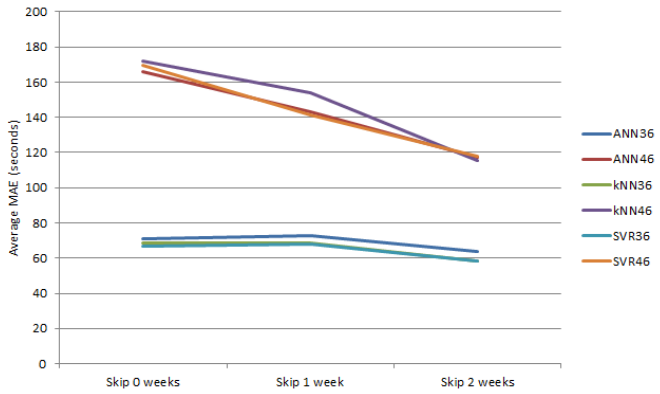

Fig. 21: The three classifiers trained on 3 weeks, and tested on the week following immediately, 1 and 2 weeks later.

APPENDIX E
K-NEAREST NEIGHBOR ATTRIBUTE ANALYSIS PLOTS

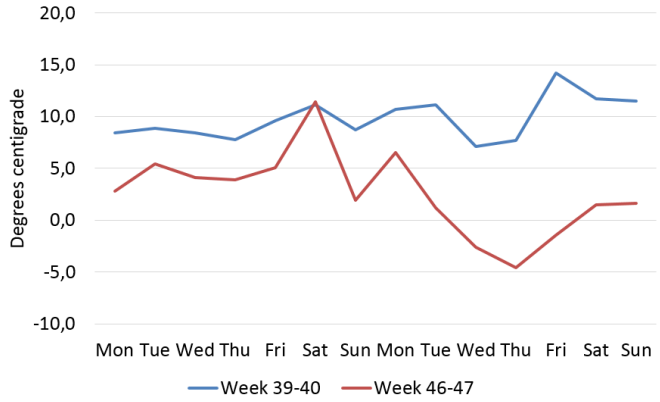

Fig. 22: Weather for weeks 39-40 and 46-47.

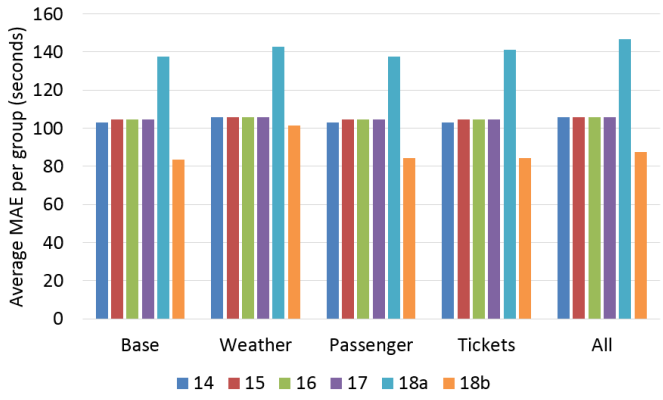

Fig. 23: Training kNN for 1 month and longer. Legend numbers depicts the dataset number. See Table 1 for dataset description.

APPENDIX F
SUPPORT VECTOR REGRESSSION ATTRIBUTE ANALYSIS PLOTS

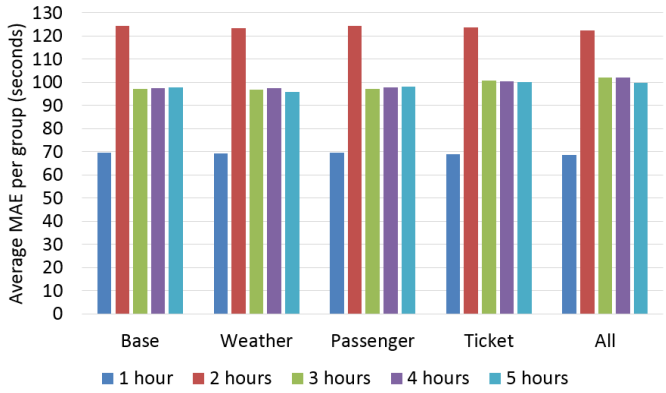

Fig. 24: Training SVR for 1-5 hours and testing on the following hour.

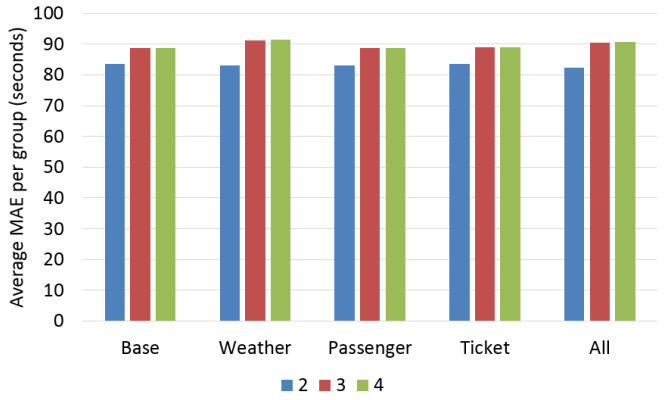

Fig. 25: Training SVR for days. Legend numbers depicts the dataset number. See Table 1 for dataset description.

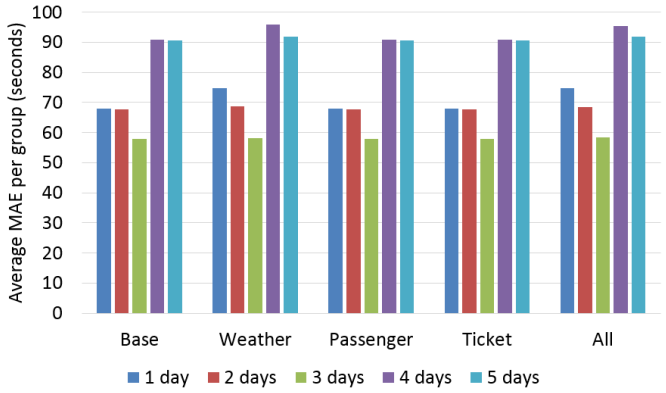

Fig. 26: Training SVR on 1-5 consecutive recurring days and tested on the next, for example trained on 3 Wednesdays and tested on the 4th.

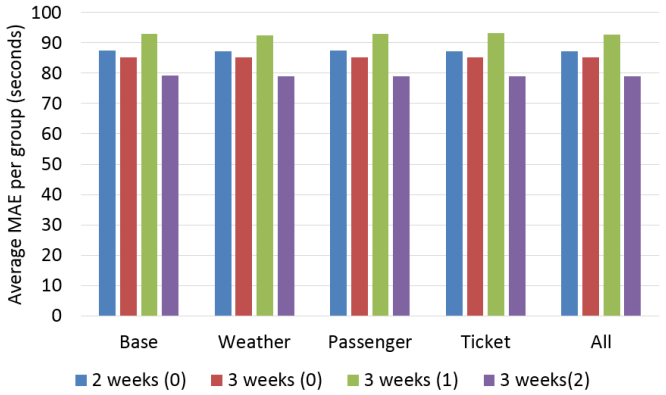

Fig. 27: Training SVR depicted number of weeks. *2 weeks (0)* means training on 2 weeks, with 0 weeks between training and test period. Testing always done on one week.

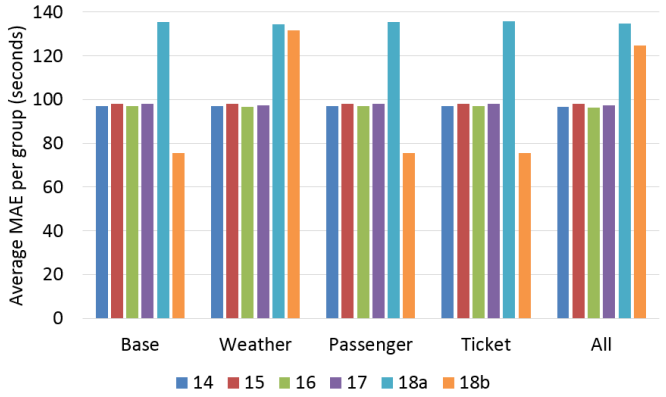

Fig. 28: Training SVR for 1 month and longer. Legend numbers depicts the dataset number. See Table 1 for dataset description.

APPENDIX G
CORRELATIONS OF TRAVEL TIMES AND DELAYS

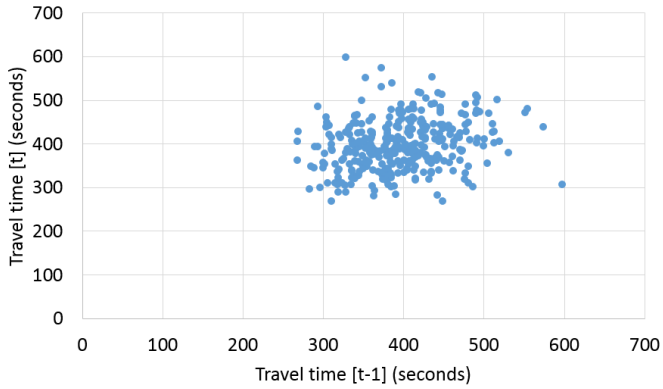

Fig. 29: Travel times for consecutive trips, *t-1* and *t*, to bus stop *Lerkendal*, 17.02.2014-21.02.2014. Correlation: 0.231

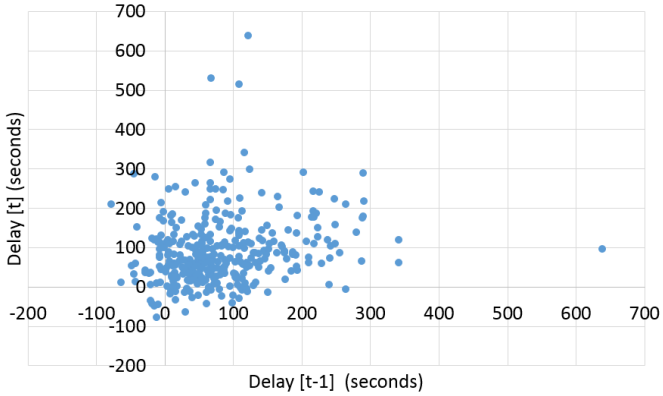

Fig. 30: Delays for consecutive trips, *t-1* and *t*, to bus stop *Lerkendal*, 17.02.2014-21.02.2014. Correlation: 0.189

APPENDIX H
PROOF OF CONSEPT RESULTS

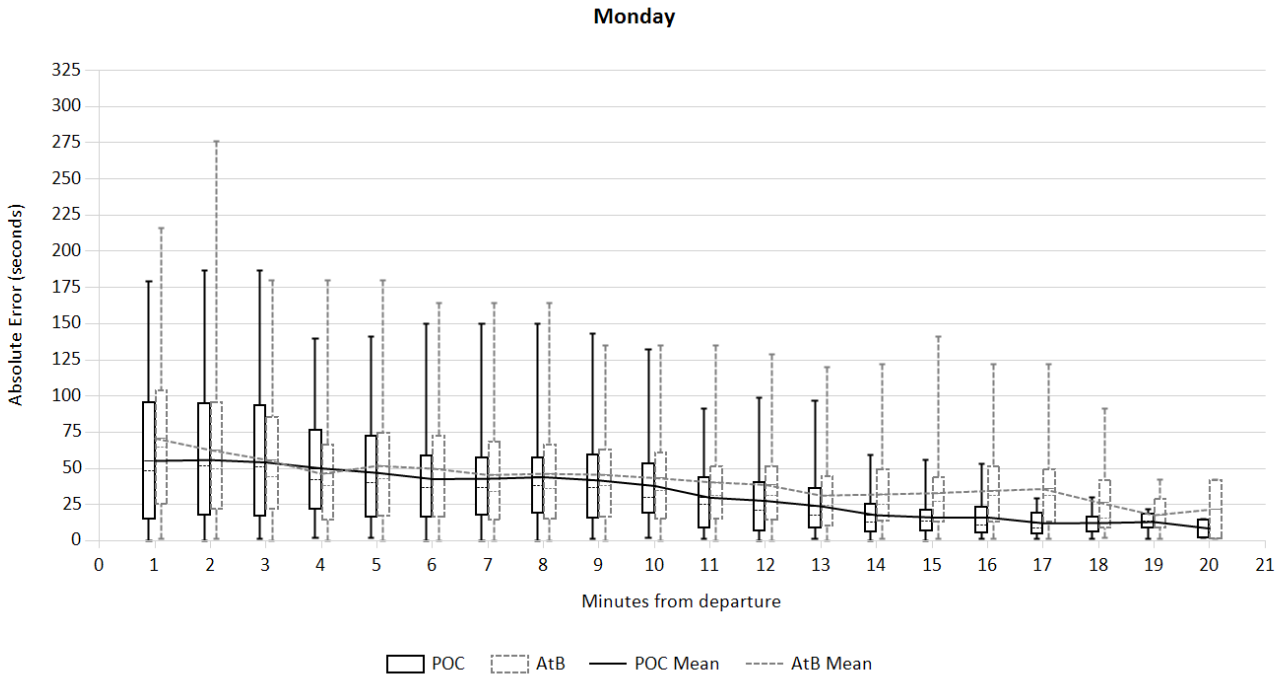

Fig. 31: Prediction errors on arrival times at *Risvollan Senter* on trips from *Kongens Gate K2*, monday 17.02.2014

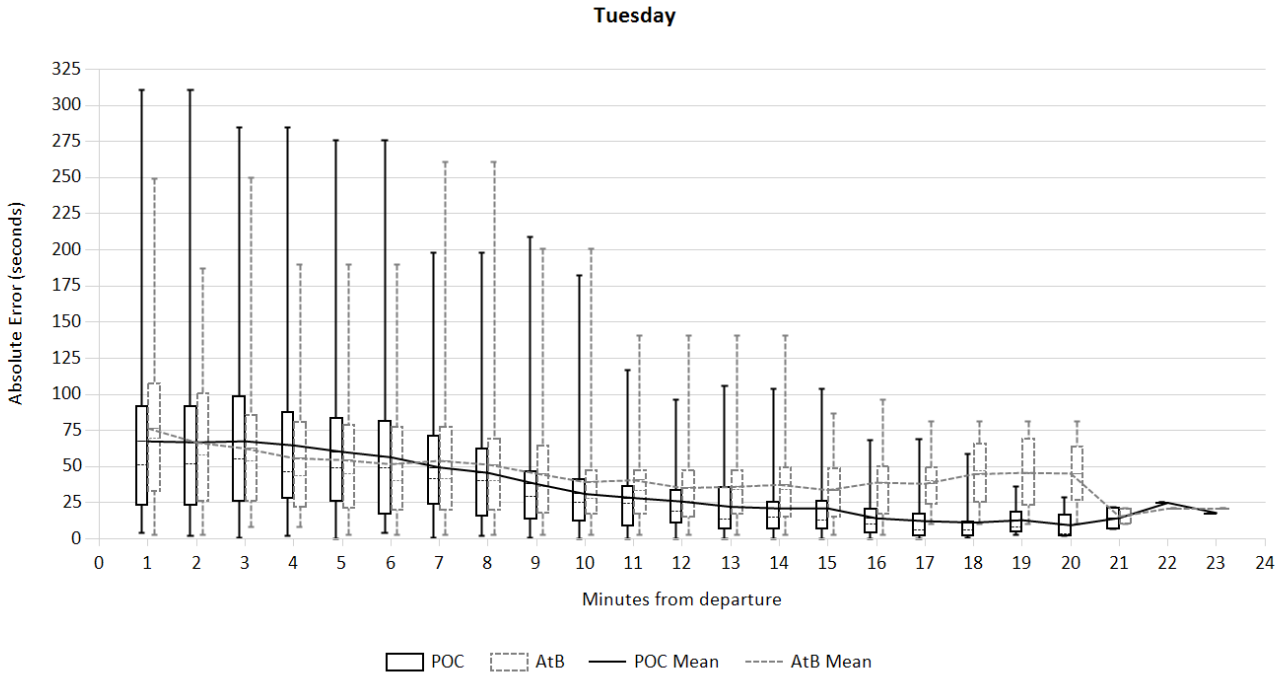

Fig. 32: Prediction errors on arrival times at *Risvollan Senter* on trips from *Kongens Gate K2*, tuesday 18.02.2014

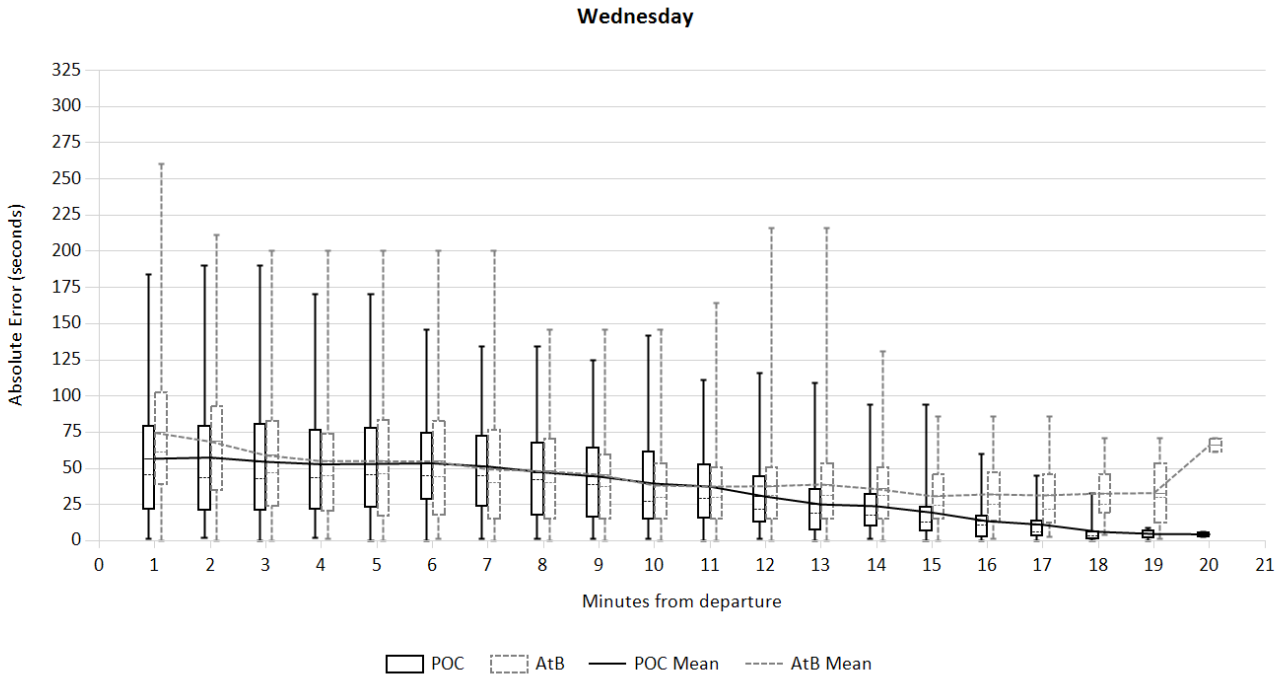

Fig. 33: Prediction errors on arrival times at *Risvollan Senter* on trips from *Kongens Gate K2*, wednesday 19.02.2014

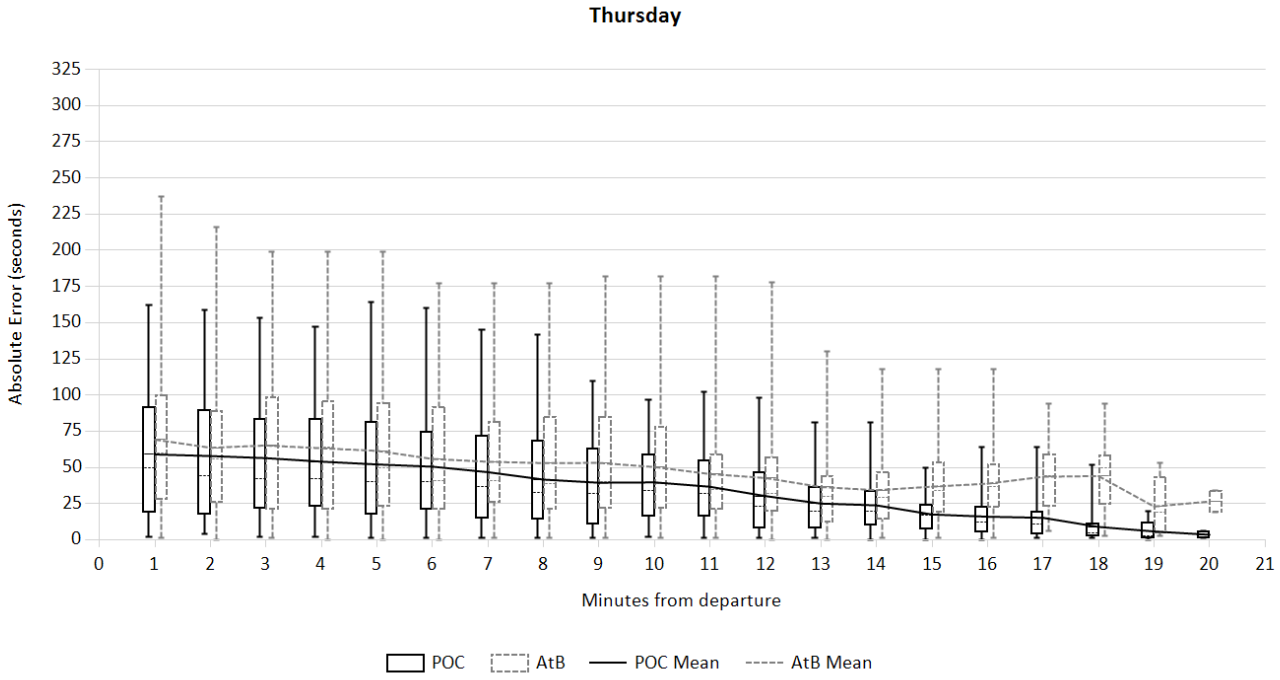

Fig. 34: Prediction errors on arrival times at *Risvollan Senter* on trips from *Kongens Gate K2*, thursday 20.02.2014

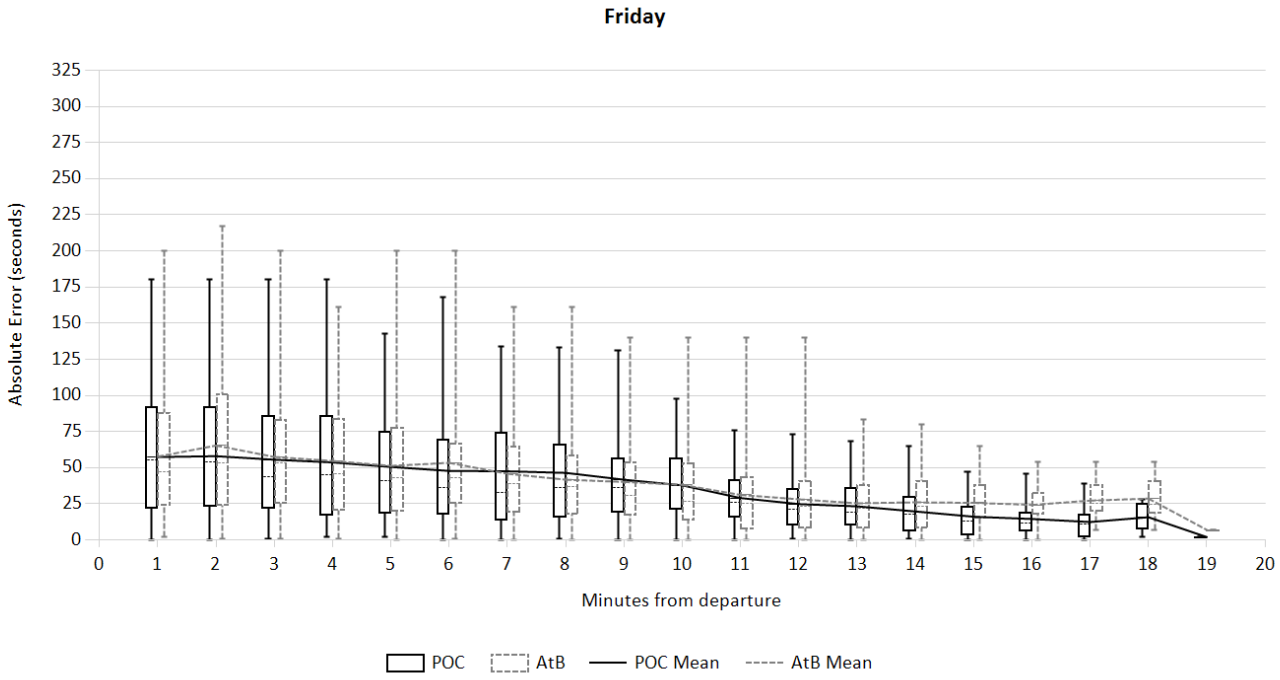

Fig. 35: Prediction errors on arrival times at *Risvollan Senter* on trips from *Kongens Gate K2*, friday 21.02.2014

APPENDIX I
STATE OF THE ART REVIEW

Intelligent Transportation Systems and Artificial Intelligence – a State of the Art Review

Aleksander Sjåfjell, Erlend Dahl and Simen Skogen

Abstract—Travel time prediction is an important part of intelligent transportation systems. The purpose of this work is to provide a general overview of the most important AI methods in the traffic domain, focusing on travel time prediction for public transport in urban areas. An extensive literature review is presented, which reveals that the most frequently applied prediction methods are k-nearest neighbor, artificial neural networks, support vector machines, Kalman filters, and some general statistics. Despite minor limitations in the methods, they are shown to outperform baseline models applied in real-world applications. The search procedures in the literature review are explained in detail, and the following section gives an introduction to each of the methods and how they are applied in the current literature. At the end of this section, the different areas of applications observed among these methods are briefly discussed. The different environments and data sets used in the literature complicate the comparison between the methods. Therefore, the final section consists of a brief summary of important features, a discussion of general trends observed in the literature, and some suggestions for future work. No methods were found to consistently outperform the others.

Index Terms—state of the art, intelligent transportation system, prediction, support vector machine, kalman filter, k-nearest neighbor, artificial neural network, machine learning

1 Introduction

Intelligent transportation systems (ITS) are advanced transportation systems which aim to apply information technology to provide improved information services, simplified management, and smoother traffic flow. Since the early 2000s the advancements in information and communication technologies have made it possible for transport systems to become more intelligent, efficient, safe and eco-friendly. Car navigation, automatic vehicle location, inductive loop detection, variable message signs, and speed cameras are examples of ITS technologies that are already extensively used worldwide.

Methods from the field of *artificial intelligence (AI)*, machine learning and statistics have been applied to ITS-collected data in a progressively increasing extent. These methods are typically categorized in search, optimization, classification, or a combination of these. In the early days of ITS, classical AI-algorithms were primarily applied for search and optimization, such as path-finding in car navigation. Due to the vast amount of higher-quality data available nowadays, ITS has allowed AI to be applied to predictions. Therefore, the cutting edge applications and research combining ITS and AI focus predominantly on traffic variable prediction.

Providing travel and traffic information to transport users anywhere and at any time will improve the efficiency and security of transport, ease congestion, and increase public transit ridership, benefiting the users and the environment. Consequently, the prediction of travel time has become a vital and prominent part of ITS applications and AI research.

This work presents a *state of the art (STAR)* literature review on methods for predicting traffic variables through available intelligent transportation systems, with special focus on travel time for buses. Firstly, section 2 presents the structured literature search, in which the choice of reviewed articles is substantiated. Secondly, section 3 gives an introduction to prediction in general. The most common methods, including kNN, ANN, SVM, Kalman filter and general statistics, are then briefly explained, together with more details on how they are implemented in the literature. Finally, in section 4 the

applied methods are compared and discussed. On the basis of this discussion, suggestions for future work are proposed.

2 The Literature Review

To retrieve as much information about the combination of AI and ITS as possible, the literature search needed to be very broad. A search query was created, but the results showed that it was impossible to create one that included all relevant parameters without returning to many articles.

The goal of this work was to get a better understanding of earlier works related to buses and travel time prediction. The chosen approach was to create three search queries; one that focused on AI methods, one for ITS, and one for travel time prediction. In addition, we also included an ITS conference (ITS Europe 2013), as this would include many of the newest articles in the field.

The search queries were executed in the IEEEXplore Command Search-utility [1], and were as follows:

> **Search 1:** *(vehicle OR bus OR train OR tram OR subway OR airplane) AND ("intelligent transportation system" OR "smart city") AND (planning OR estimation OR coordination OR "big data" OR prediction OR "travel time")*
>
> **Search 2:** *(transport) AND (planning OR estimation OR coordination OR "big data" OR prediction OR "travel time") AND ("artificial intelligence" OR svm OR svr OR "ant colony" OR id3 OR cbr OR "neural network")*
>
> **Search 3:** *((transport* OR apts OR "intelligent") AND (estimation OR prediction OR regression) AND ("travel time"))*

For more information on the search syntax, please see ref. [1]. It is important to note that we have only included results from the year 2000 and onwards. At the time of the search, October 11, 2013, a total of 837 articles were obtained (196, 175, 241, 225 for *Search 1*, *Search 2*, *Search 3* and the *ITS Conference*, respectively).

To narrow down the set of articles the reduction process was done in three steps:

1) Filtering by title text.
2) Filtering by abstract.
3) Filtering by full article.

2.1 Filtering by title text

An article was included and brought on to the next phase if the title gave an indication that the article included one or more of the following items:

1) Predicts or models transport variables.
2) Addresses a solution for a transport problem.
3) Increases the efficiency and/or adaptability of bus traffic.
4) Gives more insight in the ITS domain, or how problems in the domain are solved.
5) Has at least one of the keywords: bus, ITS, real-time or travel time prediction.
6) Mentions what it tries to solve.
7) Uses a relevant algorithm which can be utilized in public transport.
8) Is a STAR article in transportation or other similar field.

The articles were divided into three groups, where one person was responsible for filtering each group. A calibration was done on the first 25 articles of each of the groups to ensure that the inclusion criteria were interpreted in the same manner. After filtering on these criteria, a subset of 260 articles remained.

2.2 Filtering by abstract

In the second phase, the articles were filtered by considering the abstracts. The criteria weights, decided by the criterium's perceived importance, are listed in parentheses at the end of each criterium. The new quality criteria are as follows:

1) The number of times, n, an article has been selected in the searches (since multiple articles were returned by more than one search) gives the number of starting points. Articles from the ITS Conference are given three points by default, as they do not get the benefit of being found in the searches. $(0.5 \cdot n)$
2) The problem is within the transport domain. (1)

3) The solution uses an algorithm that can be connected directly, or by some modifications, to the transport domain. (2)
4) The result is clearly conveyed and can be reproduced. (0.67)
5) The solution worked like intended. (0.33)
6) The problem is relevant for traffic modeling. (2.5)
7) A STAR article is unable to get some of the points, and is awarded 4 points as compensation. (4)
8) The article is understandable. (1)

The points per article were counted, and a reasonable point cutoff was found at seven points by considering the relevance of articles above and below this line. All articles with fewer points than this were excluded. This incidentally resulted in exactly 100 articles, a reasonable amount to include in the final filtering step.

2.3 Filtering by full article

Entering the third and final stage, the articles was to be read more thoroughly. The result was the following three quality criteria, which were given points on a scale from one to five, based on the guidelines under each criterium.

1) The reproducibility of the presented solution was important, and major features were:
   a) Extensive and good descriptions.
   b) Good examples.
   c) Understandable results.
   d) Not too specific on something irrelevant.
2) The perceived relevance of the article was assessed by these features:
   a) Available data.
   b) Available tools.
   c) Is in urban environments.
   d) Is in some way connected to public transport by bus.
   e) The project described must be within reasonable boundaries regarding implementation time.
3) Is a status article. All articles being STAR articles or case studies that contributes with good and usable information.

Regular articles were rewarded points for criteria 1 and 2, and status articles for criteria 2 and 3. Note that the criteria were vague by decision, and some individual judgments were necessary to avoid excluding potential relevant articles. Finally, the point cutoff was set to seven points to catch the most relevant articles, giving the final set of 38 articles listed in the appendix.

3 State of the Art

3.1 Introduction

When predicting traffic variables, it is necessary to collect enough historical data to train the methods. Such data may include arrival times for buses, earlier travel times, measured speeds, or the level of congestion on road segments. As an example, the congestion and arrival time at previous bus stops may be used as input to predict the arrival time at the current stop. These types of data are typically stored and updated continuously in intelligent transport systems.

The aggregated data is split into a training set and a test set, and the training consists of the learning method finding a function that maps the input/output pairs in the training set. Next, the method is evaluated on the test set, and it is given feedback on whether the predictions are correct or not. This explicit feedback forms the basis of supervised learning, in contrast to unsupervised learning, where no feedback is supplied. In practice, however, the learning is semi-supervised, since both noise and lack of outputs in ITS-data create a continuum between supervised and unsupervised learning. [32]

The most common prediction methods in the reviewed literature are to be briefly explained and investigated in more detail in this chapter. These include k-nearest neighbors, artificial neural networks, support vector machines, Kalman filtering and general statistics. While statistics in general is not considered to involve AI or machine learning, this review confirms its strong presence in the ITS domain.

TABLE 1
The most common methods and applications found by the litterature review.

	Environment			Prediction goal	
	Freeway	**Urban**	**Bus**	**Travel time prediction**	**Traffic variable prediction**
kNN	2008 Zou [2], 2011 Simroth [3]	2012 Guo [4]	2012 Liu [5], 2012 Baptista [6]	2008 Zou [2], 2011 Simroth [3], 2012 Guo [4], 2012 Liu [5], 2012 Baptista [6]	
ANN	2004 Vanajakshi [7], 2008 Zou [2]	2008 Zhipeng [8], 2009 Li [9], 2009 Liu [10], 2010 Su [11], 2011 Hinsbergen [12]	2004 Jeong [13]	2004 Jeong [13], 2008 Zhipeng [8], 2008 Zou [2], 2009 Liu [10], 2009 Li [9], 2010 Su [11], 2011 Hinsbergen [12]	2004 Vanajakshi [7], 2010 Zhengxiang [14], 2011 Zheng [15]
SVM	2003 Wu [16], 2004 Vanajakshi [7]		2009 Wang [17], 2009 Peng [18]	2003 Wu [16], 2004 Vanajakshi [7], 2009 Wang [17], 2009 Peng [18]	
Kalman	2008 Jula [19]	2005 Yang [20], 2006 Liu [21], 2009 Zhu [22]	2010 Padmanaban [23]	2005 Yang [20], 2006 Liu [21], 2008 Jula [19], 2009 Zhu [22], 2010 Padmanaban [23]	
Statistics	2002 Li [24], 2011 Simroth [3], 2011 Zhu [25]	2004 Sun [26], 2010 Lin [27], 2012 Lan [28], 2012 Hofleitner [29], 2012 Hadachi [30], 2013 Gong [31]	2011 Zhu [25], 2013 Gong [31]	2002 Li [24], 2011 Zhu [25], 2012 Hadachi [30], 2012 Hofleitner [29], 2013 Gong [31]	2004 Sun [26], 2010 Lin [27], 2012 Lan [28]

3.2 k-Nearest Neighbor

k-Nearest Neighbor (kNN) is an instance based learning algorithm. In contrast to learning methods that construct a general, explicit description of the target function when training examples are provided, instance based learning methods simply store the training examples and use them directly to classify later examples [33].

The training step of the kNN algorithm is therefore just saving the training examples to a database. It is assumed that all training examples correspond to points in an n-dimensional space, and that the position of the examples in the space is based on the values of their features.

The classification is done by looking up the k nearest neighbors and using their feature values to classify the current example. How to find the nearest neighbors and how to use their values to decide the current classification are two separate problems. A typical similarity function used to select the k nearest neighbors, is to use the Euclidian distance. A typical classification function is to average the chosen neighbors' classifications. [33]

One of the reasons kNN performs so well compared to its simplicity is because it does not require any analysis of the data. A major drawback is that it will underperform when used with noisy data. An approach used to tackle this is to include a preprocessing stage.

For noisy data, Singular Spectrum Analysis (SSA) can be a solution. SSA is a technique that can extract a noise series from a plot, and leave a smoothed trend curve that is a better input for kNN. The noise part of the data is analyzed to give an estimated noise, and is then combined with the estimation from kNN to give a final prediction.

For missing data points, for example a sequential GPS series with some missing locations, interpolation can help reconstruct the route [5]. In addition to noise and flaws, GPS data are not equally spaced neither in time nor in space, which can make comparisons difficult. Interpolation has been used to be able to compare travel time on every point of the route [6].

Ref. [6] proposes the following improvements to the generic kNN algorithm:

- An exponential sliding window for giving more weight to the most recent values.
- A correction factor to adjust historical data to increase similarity or reduce distance to current time series.
- Weigh the nearest neighbors more than the distant ones.
- Error correction between the observed ground truth and the prediction value.

The sliding window was shown to not offer any significant improvements on the prediction, and was removed from future tests. The other three suggestions, however, gave a consistently better prediction in all cases (error decreased by up to 30%), independent of the chosen value of k.

The accuracy of a kNN implementation is dependent on a well-chosen value of k. The most common value of k in the traffic domain seems to be around 10 [5] [6], but ref. [4] chooses a value of 30.

Another important factor when utilizing a kNN method is the distance measurement. It is very common to utilize the Euclidian distance between the examples [4] [6], although some choose to use variants with clustering [5] or upper and lower feature bounds [2].

Finally, the method for extracting a good prediction from the chosen neighbors must be considered. Most current studies use a weighted average of the neighbors classification [4] [5] [6].

Ref. [5] compared their kNN implementation to a baseline artificial neural network, and the results indicated an improvement over the neural network in prediction accuracy. The topology of the implemented neural network conflicts with general guidelines, as the size of the hidden layer is more than twice of the input layer [34]. Consequently, the neural network may easily overfit the data, leading to lower prediction accuracy. Hence, the results may not hold in general. It was also shown that the kNN's accuracy improved as the historical data accumulated, since this made it easier to find a matching cluster for the input examples. This had the added benefit that it reduced redundancy and improved prediction speed [5]. Ref. [4] further improved kNN with the SSA preprocessing step, which made it perform better than support vector regression under both normal and incident conditions.

3.3 Artificial Neural Networks

Artificial neural networks (ANNs) are simple abstractions of the neural networks of a human brain, and consist of interconnected neurons, or nodes. The network is adapted for computation by using input nodes that are fed with parameters for the specific task to be solved, and output nodes that provide a solution. In between the input and output nodes there are one or more layers of inner nodes. An activation function converts input signals to an output signal for every node in the network. It is worth mentioning that the input nodes serve the inputs for the environment, normally without alternation. Hence, the identity function is employed in the input layer.

In feed-forward neural networks, the signals propagate in one direction only – from input to output. However, in recurrent networks the connections can be directed backwards to previous layers or to nodes in the same layer. Feed-forward networks are most commonly used, as they are simpler to implement and analyze than recurrent networks. Some of the connections are more important (i.e., stronger weighted) than others for producing the correct output. The optimal weights on the connections are often learned in a supervised fashion by training on samples that we know the solution to. To minimize the error (amount of incorrect classifications) during training, the backpropagation algorithm presented by ref. [35] is the most commonly used.

ANNs are popular due to their ability in finding nonlinear patterns in complex problems, purely based on the data. However, the process of training is typically very time-consuming compared to other methods. In addition, the network topology (the amount of nodes and their connections), the learning rule of adjusting the weights, and the activation functions are interdependent; they need to fit together. As an example, backpropagation requires the activation function to be differentiable. This interdependency means that there

are many variations of ANNs suited for different problems.

Although the nodes have very limited computing capabilities, they can perform complicated tasks when connected together. The collective emergent behavior is difficult to interpret, as the knowledge during training is stored in an implicit manner. The behavior of ANNs are therefore often referred to as a ”black box” [7]. Since there are no general rule on how to structure the ANN for specific problems, the optimal structure (the number of input notes, layers and hidden neurons) is typically found empirically, rather than analytically [7] [13] [15].

ANNs have proven to be competitive to other methods for the prediction of traffic parameters. The optimal structure of an ANN is highly dependent on the specific problem, which has led to many specialized variations of the general feed-forward ANN described here. However, the general multi-layered feed-forward ANN is still much applied and compared to other methods for traffic prediction [7] [10] [13] [15].

Ref. [13] applies ANN for bus arrival time prediction by incorporating dwell time, schedule adherence and arrival time for each stop as input variables. The number of hidden neurons, in addition to the training and learning functions, are chosen through empirical analysis. The ANN clearly outperforms the implemented models based on linear regression and historical averaging. However, the inclusion of real time schedule adherence data did not improve the ANN-predictor noteworthy. They hypothesized that this outcome was caused by the nonlinear relationship between arrival time and schedule adherence, despite the capabilities of ANNs to solve nonlinear problems in general. In ref. [7] and ref. [13], the networks output predictions of the same variable type that is used for input, namely traffic speed and arrival time. However, ref. [10] argues that the dynamic change of such a variable is due to external factors, and that it has no causal relationship to itself. In response, they propose an indirect approach, using other traffic quantities than the variable to be predicted as input.

Combining prediction models by taking a weighted average of their predictions is a well -established approach. It has been shown that such an ensemble or committee approach cannot lead to increased prediction error; committees of neural networks in special, have shown accurate predictions [12]. In this study, 230 ANNs with different structures are combined in a committee and applied to travel time prediction in urban road segments. A Bayesian framework is used to train and rank the networks for selection and combination. Despite the stochastic nature of urban travel times, the committee proved to be able to predict the underlying, low-frequency trends with accuracy very similar to the prediction of freeway travel time with the same framework. In ref. [9], the AdaBoost method is used on a committee of ANNs with the same purpose of predicting travel time. They confirm that the committee of ANNs outperforms a single network. Although combining models typically improves the accuracy, it requires significantly more computation time, both with respect to training and classification. Hence, committees may not be suitable for real-time applications.

Radial Basis Function (RBF) networks is another type of neural networks that has been applied to traffic prediction [11] [14]. These networks are trained in two phases; first unsupervised by clustering, then supervised. Since the supervised training does not require backpropagation, these networks can be more efficient in terms of speed [11] [32]. However it is not clear whether or not these networks outperform general feed-forward backpropagation ANNs in accuracy.

3.4 Support Vector Machines

Support Vector Machines (SVM) can be utilized when specific domain knowledge cannot be used, or does not exist. To illustrate how SVMs work, input data can be visualized as a two-dimensional graph, where the goal is to draw a single line to separate the data into two groups. The assumption is that if this linear separation is good, a future data point will be placed in the correct half of the space. [32]

Since few actual data sets can be separated in the two-dimensional space, SVMs use kernel

functions to map the data to a higher dimension, where it will be linearly separable. In general, a data set with n data points will be linearly separable in a space with $n-1$ or more dimensions. This means that even though the data set looks garbled in the original space, an SVM can draw a hyperplane in a higher dimension, and then transform this line back to the original space to produce a non-linear separator that can be used to classify future examples. A penalty parameter on misclassified examples is used to choose between high and low complexity functions. [32]

As they can work in higher dimensions, SVMs have the ability to represent complex functions. Since they use a maximum margin separator to classify the input data, they also generalize very well. The result of this is that only some of the data points (the support vectors) are essential to create the separator, and the SVMs are thus resistant to overfitting. [32]

Support Vector Regression (SVR) is a version of the SVM which, as the name suggests, is adapted to performing regression. This is particularly useful in traffic scenarios, since a lot of the data can be represented as time series that can be expanded by regression.

When using SVM predictors, there are two important decisions that are discussed. Firstly, one has to choose a kernel function, and secondly, one has to find the right parameters for the chosen kernel.

The four most popular kernels in the traffic literature are the linear kernel, the polynomial kernel, the RBF kernel, and the sigmoid kernel. Some studies have compared the different kernels, and have found that the Linear and RBF kernels are the best for traffic prediction [16] [17]. The latter reference proposed that RBF is the best of these, since it is the only kernel which performance does not degrade when sample size decreases. When the kernel function has been determined, its parameters have to be selected, usually by trial and error.

SVMs have been compared to some simple historical statistical predictors and to an ANN. It consistently outperformed the statistical predictors [16], and was shown to give better predictions than the ANN when they were given just a few days of training data [7]. The ANN did, however, outperform the SVM when given more data.

A proposed improvement for the SVMs used in traffic prediction is to exchange the normal penalty factor c with the parameter v. The c parameter is usually used to apply a penalty to examples that are on the wrong side of the hyperplane, while the supposedly improved v parameter sets a limit on both the number of examples that are support vectors, and the number of examples that are on the wrong side [36]. The v-SVM has been tested and proven to be reliable, but it was not compared to a standard c-SVM, which makes it difficult to assess the improvement [17].

Another suggestion is to use a relevance vector machine (RVM) instead of an SVM. The RVM is a probabilistic model that is very similar to the SVM, but it is trained under the Bayesian framework, and has a few advantages compared to the SVM; it has fewer parameters to estimate, its kernel function can be more relaxed, and the computational complexity is lower. In addition, the RVM has the advantage that it can return the uncertainty of the prediction, which could be useful in assessing the predictions. [18]

3.5 Kalman Filter

Kalman filtering is a method that can be used to predict future states when working with observations presumed to be noisy. In traffic prediction, the observations are often GPS measurements, which are known to be imprecise. They are also seldom real-time, but rather sent periodically, for example once every few minutes. The Kalman filter then calculate future state vectors, which can be used to estimate travel time or traffic volume.

A Kalman filter will use previous observations and estimations to calculate the future estimations. When a new observation arrives, weighted averages will be used to update the estimation to improve the precision of following estimations. Since the estimated state is cumulative, the filter will only need the previous estimate and the latest observation to make another estimate, which makes it very suitable for online applications.

When using the Kalman filtering technique, the most prevalent data collection method is to use passive test vehicles (e.g., coils or roadside detectors) [2] [19] [21] [22] . However, research has been done with active test vehicles too – ref. [23] uses a technique based on Kalman filtering on consecutive buses. Another small scale study was done using cars rigged with GPS equipment that drove through congested arteries with five minute intervals to probe the traffic status. [20].

How often to collect data is a difficult problem that requires careful assessment. A high frequency of data collection will give higher prediction accuracy up to a certain point, but more data has to be stored and more processing power will be needed to analyze it. Having the logging frequency too low can result in losing important traffic trends, and having it to high can make the method run too slow for real-time usage. Multiple studies have found that a frequency in the range of 2-5 minutes is a good compromise [19] [20] [21] [22].

When considering urban traffic, a challenge is how to do calculations when roads influence each other at intersections. One article added link relations to the Kalman filter to handle this, and got a very effective predictor that got half the MAPE of a simple time series prediction method, but was unfortunately not compared to other Kalman models [22]. Another study on Kalman filtering halved their MARE by adding slope adjustment and two-point data interpolation [20]. Ref. [21] tried to train an ANN with a Kalman filter, and found that this combination performed better when given noisy data than a state space model solved by a Kalman filter.

3.6 Statistical Methods

Several of the reviewed articles employ simple statistical methods in order to have a baseline method to compare with their supposedly improved methods. A few articles, however, have created stand-alone statistical methods in attempts to improve the predictions, and these will be discussed briefly in this section.

A very simple statistical approach that has been used in practice, is to calculate the average speed for travel segments based on the GPS positions of buses, and on historical data if the real-time data is faulty or missing [37]. A slightly more complex method is a Markov Chain with a Gaussian Mixture model and Monte Carlo integration that is used to do better predictions in situations with missing data. This method was shown to outperform simple statistical methods, such as historical average, regular Markov chains, and combinations of these [26].

Another example is a small scale study where GPS data from three sequential buses were used to predict the travel time of the last of the three [38]. This approach was later improved so that it did not need data from all three vehicles, and made more suitable for real-time applications [23]. This method works by analyzing the GPS data, and tries to predict running time (the time actually spent in movement) and delay time (delay as a result of traffic conditions) in order to get an overall prediction. A similar technique was utilized in another study, where moving average was used to predict running time and dwell time (the time standing still at bus stops).

A slightly newer study calculates bus travel time by simply using average speed from the GPS if the destination is one stop ahead, since the chances of traffic condition changes are small on such short distances. When the bus is more than two stops ahead, the prediction is divided into running time, dwell time and delay time – a small addition to the previously mentioned studies, which did not consider all of these separately. These three predictions are combined to give a final prediction on a per link basis and then summed up across the entire distance. This method gave promising results, and it is interesting to note that the dwell time seems to be easiest to predict, with the two other having nearly twice as high MAPE. [25]

3.7 Areas of application

As table 1 shows, the papers are partitioned into three different environments, and two different prediction goals. A paper normally appears once in one of the three first columns, and once in the two last.

The papers about highway prediction typically report good results with simple methods, while the methods will need higher complexity in urban environments. This is because of the added complexity given by typical urban obstacles such as road segments meeting in intersections, stop lights, and the queues these result in. The bus predictions are further complicated by the time the bus has to spend at bus stops, loading and unloading passengers.

The methods mentioned in section 3 are the most common returned by the literature review. These were mostly used for travel time predictions, but there were some papers about general traffic status prediction, and a few papers covering other areas, such as passenger flow or supply capacity. The articles covering other areas than travel time predictions are grouped together in the rightmost column.

When dealing with bus travel times, a good approach seems to be to partition the overall prediction into multiple parts, considering the different movement patterns of a bus. Several articles separate the travel times into running time and non-running time [31] [38], while other partition it further by splitting the non-running time into time spent on bus stops, and time spent waiting in traffic [6] [25]. As the bus stops frequently, and spends a significant part of the overall travel time at bus stops, partitioning the prediction time is reasonable, and gives better overall results.

The other areas of application are usually attempts to estimate the traffic flow on road segments. The most common is to estimate the traffic speed [7] [27] [28], while the last few papers attempt to estimate other features that can tell us something about the traffic flow, which includes transport volume [14], vehicles per hour [26] and passenger count on a railway [15].

4 Conclusion and suggestions for future work

There are four commonly used methods in the current traffic literature: kNN, ANN, SVM and Kalman filters. All of these methods are capable of making travel time predictions better than baseline models, either on their own or combined. Since most articles apply methods to different data sets, it is difficult to compare them on a fair basis. Hence, comparisons are only summarized from articles that explicitly focus on this. More importantly is the analysis of each of the applied methods.

Despite the simplicity of kNNs in general, improvements have been proposed. Firstly, kNN achieved higher accuracy by utilizing preprocessing steps like SSA, in addition to the extension of a correction factor on chosen neighbors and error correction on predicted values. Secondly, ref. [2] demonstrated good results with a combination of kNN and ANN.

When improved with SSA, the kNN method implemented by ref. [4] outperformed an SVR implementation, almost halving its error. This is a promising result for the kNN method, but in lack of comparisons against the cutting-edge implementations of ANN and SVR models found in the current literature, it is difficult to draw any conclusions.

ANNs are still much applied without substantial specialization. Although committees of ANNs have shown promising results, they may not be suitable for real-time applications, due to the extensive computation time and resource requirements. RBF networks are faster learners and may be better suited for such applications. However, more research on the prediction accuracy of RBF networks in comparison to feed-forward backpropagation networks is needed in order to evaluate its payoffs.

The SVM implementations in the current literature are also very similar, with only a few articles selecting other kernel functions than the usual RBF kernel. Two promising implementation variations, v-SVM and RVM, have been proposed and shown to be reliable, but neither of them has been compared to the original SVM to show an actual improvement.

The predictions of Kalman filters have been improved with slope adjustments and interpolations. Although unscented Kalman filters have been proven to consistently outperform extended Kalman filters [39], no implementations are found in the literature. Hence, future research should evaluate the potential of the unscented variant in the ITS domain.

A more thorough comparison between

ANNs and SVMs was performed by ref. [7]. Their results indicated that with less training, the implemented SVM would outperform the ANN, while the ANN would outperform the SVM with more training. They argued that since the SVM is independent of the training data once the support vectors are chosen, more training data will not necessarily induce a lower error.

The optimal structure and choice of parameters are highly dependent on the specific problem for all applied methods. This leads to a lot of trial and error approaches to decide important parameters, such as the value of k for kNN, the network topology for ANNs, and the kernel function with its parameters for SVMs. This way of determining optimal values is time consuming, and future work should focus on finding alternative ways where possible.

kNN methods, Kalman filters and SVMs are possibly better suited for doing real-time traffic predictions than ANNs, since the nature of the latter makes it necessary to train it all over again to add any recent observations from the traffic. Nevertheless, the ANNs have shown potential, and may be trained less frequently while still being applicable for real-time predictions.

The research on travel time prediction in urban environments, which has been on the rise the recent years, is heavily influenced by knowledge from previous research on travel time prediction in single links, such as freeway sections. However, due to the stochastic nature of urban environments, predicting travel time accurately has been proven hard. This difficulty is typically reinforced by the limited amount of different data sources applied; the prediction models are sometimes based solely on historical travel times.

Specifically for bus travel time prediction in urban environments, partitioning the travel time into running time, dwell time and delay time has given good results. This makes intuitive sense, as the three parts are not necessarily interdependent. In addition, experiments considering relations between connected road segments have been performed with promising results. The segment relations may be vital for further research.

Although travel time distributions show periodic trends on a daily basis, travel time has no causal relationship to itself; subsequent travel times are not dependent on each other, but rather on external factors. Therefore, one should include other data sources that are hypothesized to influence the travel time, such as weather data or information about large events. In general, travel time prediction should start moving towards the science of big data in order to handle the increasing amount of available and potentially relevant input data.

Appendix
The Final Set of Articles with Scores

Table A1 lists the final set of articles and their score.

Acknowledgments

The authors would like to thank Jo Skjermo for being an exceptional mentor, Tomas Levin for contributing with his domain knowledge, and Anders 'Method-is-my-friend' Kofod-Petersen for structuring the work when it was most needed.

References

[1] IEEEXplore. (2013) IEEEXplore command search. [Online]. Available: http://ieeexplore.ieee.org/search/advsearch.jsp?expression-builder

[2] N. Zou, J. Wang, and G.-L. Chang, "A reliable hybrid prediction model for real-time travel time prediction with widely spaced detectors," in *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, 2008, pp. 91–96.

[3] A. Simroth and H. Zähle, "Travel time prediction using floating car data applied to logistics planning," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 1, pp. 243–253, 2011.

[4] F. Guo, R. Krishnan, and J. W. Polak, "Short-term traffic prediction under normal and incident conditions using singular spectrum analysis and the k-nearest neighbour method," in *Road Transport Information and Control (RTIC 2012), IET and ITS Conference on*, 2012, pp. 1–6.

[5] T. Liu, J. Ma, W. Guan, Y. Song, and H. Niu, "Bus arrival time prediction based on the k-nearest neighbor method," in *Computational Sciences and Optimization (CSO), 2012 Fifth International Joint Conference on*, 2012, pp. 480–483.

[6] A. Baptista, E. Bouillet, and P. Pompey, "Towards an uncertainty aware short-term travel time prediction using gps bus data: Case study in dublin," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, 2012, pp. 1620–1625.

TABLE A1
The final 38 articles this work is based on.

Article	Score
2013 Zhang [40]	10
2008 Zhipeng [8]	10
2012 Baptista [6]	10
2009 Peng [18]	10
2013 Gong [31]	10
2009 Zhu [22]	9
2009 Sananmongkhonchai [41]	9
2012 Liu [5]	9
2010 Padmanaban [23]	9
2011 Zhu [25]	9
2004 Jeong [13]	9
2007 Li [42]	8
2011 Zheng [15]	8
2009 Liu [10]	8
2010 Su [11]	8
2011 van Hinsbergen [12]	8
2009 Li [9]	8
2012 Hadachi [30]	8
2009 Padmanaban [38]	8
2009 Wang [17]	8
2004 Manolis [37]	7
2012 Guo [4]	7
2010 Zhengxiang [14]	7
2010 Lin [27]	7
2004 Vanajakshi [7]	7
2006 Zidi [43]	7
2003 Wu [16]	7
2012 Lan [28]	7
2005 Yang [20]	7
2006 Liu [21]	7
2011 Simroth [3]	7
2012 Hofleitner [29]	7
2002 Li [24]	7
2008 Jula [19]	7
2004 Sun [26]	7
2008 Zou [2]	7
2012 Ernst [44]	7
2009 Vanajakshi [45]	7

[7] L. Vanajakshi and L. Rilett, "A comparison of the performance of artificial neural networks and support vector machines for the prediction of traffic speed," in *Intelligent Vehicles Symposium, 2004 IEEE*, 2004, pp. 194–199.

[8] Z. Li, N. Li, F. Liu, and Y. Liu, "Short-term forecasting of travel time based on license plate matching data," in *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, 2008, pp. 1390–1395.

[9] Y. Li, R. Fujimoto, and M. Hunter, "Online travel time prediction based on boosting," in *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on*, 2009, pp. 1–6.

[10] H. Liu, K. Zhang, R. He, and J. Li, "A neural network model for travel time prediction," in *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, vol. 1, 2009, pp. 752–756.

[11] H. Su, Y. Hu, and J. Xu, "Travel time estimating algorithms based on fuzzy radial basis function neural networks," in *Control and Decision Conference (CCDC), 2010 Chinese*, 2010, pp. 2653–2657.

[12] C. van Hinsbergen, A. Hegyi, J. van Lint, and H. van Zuylen, "Bayesian neural networks for the prediction of stochastic travel times in urban networks," *Intelligent Transport Systems, IET*, vol. 5, no. 4, pp. 259–265, 2011.

[13] R. Jeong and L. Rilett, "Bus arrival time prediction using artificial neural network model," in *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, 2004, pp. 988–993.

[14] Y. Zhengxiang, X. Guimin, and W. Jinwen, "Transport volume forecast based on grnn network," in *Future Computer and Communication (ICFCC), 2010 2nd International Conference on*, vol. 3, 2010, pp. V3–629–V3–632.

[15] D. Zheng and Y. Wang, "Application of an artificial neural network on railway passenger flow prediction," in *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, vol. 1, 2011, pp. 149–152.

[16] C.-H. Wu, C.-C. Wei, D.-C. Su, M.-H. Chang, and J.-M. Ho, "Travel time prediction with support vector regression," in *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, vol. 2, 2003, pp. 1438–1442 vol.2.

[17] J. nan Wang, X. mei Chen, and S. xia Guo, "Bus travel time prediction model with v - support vector regression," in *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on*, 2009, pp. 1–6.

[18] C. Peng, Y. Xin-ping, and L. Xu-hong, "Bus travel time prediction based on relevance vector machine," in *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*, 2009, pp. 1–4.

[19] H. Jula, M. Dessouky, and P. Ioannou, "Real-time estimation of travel times along the arcs and arrival times at the nodes of dynamic stochastic networks," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, no. 1, pp. 97–110, 2008.

[20] J.-S. Yang, "Travel time prediction using the gps test vehicle and kalman filtering techniques," in *American Control Conference, 2005. Proceedings of the 2005*, 2005, pp. 2128–2133 vol. 3.

[21] H. Liu, H. van Lint, H. Van Zuylen, and K. Zhang, "Two distinct ways of using kalman filters to predict urban arterial travel time," in *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, 2006, pp. 845–850.

[22] T. Zhu, X. Kong, and W. Lv, "Large-scale travel time prediction for urban arterial roads based on kalman filter," in *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, 2009, pp. 1–5.

[23] R. Padmanaban, K. Divakar, L. Vanajakshi, and S. Subramanian, "Development of a real-time bus arrival prediction system for indian traffic conditions," *Intelligent Transport Systems, IET*, vol. 4, no. 3, pp. 189–200, 2010.

[24] Y. Li and M. McDonald, "Link travel time estimation using single gps equipped probe vehicle," in *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, 2002, pp. 932–937.

[25] T. Zhu, F. Ma, T. Ma, and C. Li, "The prediction of bus arrival time using global positioning system data and dynamic traffic information," in *Wireless and Mobile Networking Conference (WMNC), 2011 4th Joint IFIP*, 2011, pp. 1–5.

[26] S. Sun, G. Yu, and C. Zhang, "Short-term traffic flow forecasting using sampling markov chain method with incomplete data," in *Intelligent Vehicles Symposium, 2004 IEEE*, 2004, pp. 437–441.

[27] C. Lin, W. Shutao, and S. Linxiang, "Traffic status estimate based on data fusion," in *Education Technology and Computer (ICETC), 2010 2nd International Conference on*, vol. 5, 2010, pp. V5–144–V5–146.

[28] J. Lan, M. Guo, Z. Lin, J. Li, T. Aibibu, and W. Xiao, "Space matching fusion model for arterial speed estimation in its," in *Information Fusion (FUSION), 2012 15th International Conference on*, 2012, pp. 861–866.

[29] A. Hofleitner, R. Herring, P. Abbeel, and A. Bayen, "Learning the dynamics of arterial traffic from probe data using a dynamic bayesian network," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, no. 4, pp. 1679–1693, 2012.

[30] A. Hadachi, S. Mousset, and A. Bensrhair, "Practical testing application of travel time estimation using applied monte carlo method and adaptive estimation from probes," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, 2012, pp. 1078–1083.

[31] J. Gong, M. Liu, and S. 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*, 2013, pp. 972–976.

[32] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2009.

[33] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[34] R. Callan, *The Essence of Neural Networks*, ser. The Essence of Computing Series. Prentice Hall Europe, 1999. [Online]. Available: http://books.google.com.pe/books?id=uWtnQgAACAAJ

[35] D. E. Rumelhart, G. E. Hintont, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[36] P. hsuen Chen, C.-J. Lin, and B. Schölkopf, "A tutorial on v-support vector machines."

[37] K. Manolis and D. Kwstis, "Intelligent transportation systems - travelers' information systems the case of a medium size city," in *Mechatronics, 2004. ICM '04. Proceedings of the IEEE International Conference on*, 2004, pp. 200–204.

[38] R. P. S. Padmanaban, L. Vanajakshi, and S. Subramanian, "Automated delay identification for bus travel time prediction towards apts applications," in *Emerging Trends in Engineering and Technology (ICETET), 2009 2nd International Conference on*, 2009, pp. 564–569.

[39] R. V. D. Merwe and E. A. Wan, "The square-root unscented kalman filter for state and parameter-estimation," in *International Conference on Acoustics, Speech, and Signal Processing*, 2001, pp. 3461–3464. [Online]. Available: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.7711

[40] Y. Zhang, J. Tang, and S. Lv, "Least time path planning under urban timetabled public transport using a* algorithm," in *Control and Decision Conference (CCDC), 2013 25th Chinese*, 2013, pp. 3587–3592.

[41] S. Sananmongkhonchai, P. Tangamchit, and P. Pongpaibool, "Cell-based traffic estimation from multiple gps-equipped cars," in *TENCON 2009 - 2009 IEEE Region 10 Conference*, 2009, pp. 1–6.

[42] Q. Li, S. Xie, X. Tong, and G. Liu, "Path planning algorithm for vehicles based on time-dependent optimization criterion," in *Control and Automation, 2007. ICCA 2007. IEEE International Conference on*, 2007, pp. 2360–2364.

[43] S. Zidi, S. Maouche, and S. Hammadi, "Real-time route planning of the public transportation system," in *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, 2006, pp. 55–60.

[44] J. Ernst, J. Krogmeier, and D. Bullock, "Kullback-leibler comparison framework for the evaluation of travel time distribution estimates," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, 2012, pp. 564–569.

[45] L. Vanajakshi, S. Subramanian, and R. Sivanandan, "Travel time prediction under heterogeneous traffic conditions using global positioning system data from buses," *Intelligent Transport Systems, IET*, vol. 3, no. 1, pp. 1–9, 2009.