Bachelor's project

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Terje Haugum
Bjørn Kristian Punsvik
Brage Wichstrøm

Implementation and comparison of different machine learning models' ability to predict travel time of public transport

Bachelor's project in Computer Engineering
Supervisor: Ole Christian Eidheim
May 2020

NTNU
Kunnskap for en bedre verden

Terje Haugum
Bjørn Kristian Punsvik
Brage Wichstrøm

Implementation and comparison of different machine learning models' ability to predict travel time of public transport

Bachelor's project in Computer Engineering
Supervisor: Ole Christian Eidheim
May 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU

Kunnskap for en bedre verden

Abstract

In this paper we aim to utilize machine learning methods on open near real-time data to predict the arrival times of buses. The overarching task is to predict anomalies from the planned bus schedule existing in Oslo using machine learning models. Predicting arrival times is an important segment within Intelligent Transport Systems (ITS), and a daunting task in urban areas due to excessive traffic anomalies. Our main focus of inquiry is to compare a set of alternative implemented models with a simple historical model in order to identify an optimal model for predicting arrival times of buses in urban areas. We conducted a literature search on previous work on this issue, and identified the most typical models that tended to outperform the current real-world applications. Artificial neural networks(ANN) is the most frequently used model. However, Support Vector Regression(SVR) and Long-Short-Term-Memory(LSTM) models have also proven to perform well on finding correct arrival times. The literature regarding artificial models and other models considered to be relevant to the issue are explained in detail. In the section thereafter we detail where we received the data used in our models. Moreover, how we implemented our purposed models and how we sanitized the received data with data-conversions and interpolations are presented in the same section. We found LSTM to outperform the other models in regards to accuracy while ANN proved to be the most economical of the investigated models considering both time to train and accuracy.

Assignment details

Real-time data on the position of buses in several Norwegian cities is available through several agencies, e.g. *Entur*. Knowing the arrival times is important for travelers and creators of the bus service system alike. However, anecdotal evidence clearly show that planned arrival times based on historical data do not correspond to actual arrival times. We therefore aimed to compare and contrast alternative modelling procedures, with a special emphasis on the accuracy of machine learning methods. To accomplish this, we will collect, analyze and preprocess real-time data for a selected city, before feeding the data into machine learning models. We then investigate the performance of a simple model that uses only historical averages of travel times, and more advanced models using neural networks. Finally, we compare these models to determine the optimal model for solving the complex problem of finding arrival time predictions within an urban area.

The assignment was made for our bachelor thesis in computer engineering by the company Kantega in cooperation with the Norwegian University of Science and Technology(NTNU).

Special thanks

We would like to give a special thanks to Kantega who allowed us to write about this subject in cooperation with the Norwegian University of Science and Technology. They offered us a location where we could work with our study, and provided substantial knowledge on the subject. We would also like to thank our mentors Daniel Lisø from Kantega and Ole Christian Eidheim from NTNU that guided us with great insight in the process, article review surrounding this study, and last but not least machine learning knowledge.

Contents

List of Figures

List of Tables

1 Introduction

The development of intelligent transportation systems (ITS) has grown in popularity as vehicles gain insight to their environment using sensors and assistant systems. ITS is an important contributor to traffic efficiency, overall traffic flow and road safety (Festag 2014). By using the information the vehicles can produce, there is an opportunity to utilize artificial intelligence (AI) models to calculate travel times. The numerous studies conducted trying to implement intelligent models and comparing them with simple models such as historic models, makes use of the opportunity made available by ITS. We will present a couple of these studies in Section 2.

1.1 The importance of the study

With inefficient traffic comes high traffic congestion. This is an issue regarding high costs with vehicle emissions, wasted fuel and wasted time for the cities and people involved, resulting in high monetary costs as well. In 2019, a report given by The Texas A&M Transportation Institute estimated that the average commuter wasted 7 full workdays in 2017 due to traffic delays. (Schrank, Eisele, and Lomax 2019). This report also emphasizes that traffic congestion is still a growing issue. As indicated by the research of (Beaudoin, Farzin, and Lawell 2013), the detrimental effects of delays and congestion could be improved by investments in the public transportation system. Having access to correct travel and arrival times could be one such improvement.

In addition to road safety and efficiency, ITS is important for the bus agencies regarding having a reliable solution for predicting arrival times. Reliability has an important impact whether or not the customer changes their way of transportation. In a survey performed by (Carrel, Halvorsen, and Walker 2013) in San Francisco, they found the local buses to be most vulnerable, where 40% of the participants said they would use them less due to their unreliability. Moreover, the respondents conveyed that they indeed were likely to change their transportation method all together due to such unreliability. A bus not holding its schedule is one issue, arguably not knowing when it eventually will arrive could add to customer frustration. Hence it comes as no surprise that generating accurate travel time predictions for the buses increases the satisfaction to the customers (Jeong and Rilett 2004).

However, to determine the most accurate arrival times regarding each city is a daunting task. A rush hour at 07:00 in on city can mean a stable traffic flow in another city where the rush does not start until 08:00. Using models that only rely on stationary data such as speed limits and average speed do not accumulate the effect dwell times at each stops has on the forthcoming arrival time on the next stop. However, machine learning models can utilize these stochastic

factors such as dwell time to predict arrival times with better accuracy. Using systems within ITS such as Automatic Vehicle Location(AVL) and or Automated Passenger Counter (APC) that tracks the location and the number of passengers will provide the artificial intelligence models data needed to calculate accurate arrival times (Fabrikant 2019). However, the data gathered from these types of systems rely on having them installed in the vehicles gathering data. Not all bus agencies or agencies that gather data for traffic flow/safety improvement have these systems installed.

1.2 Our study

In this inquiry we will investigate three artificial intelligence models highly capable of finding arrival time prediction using Real time data, historical data and data gathered from ITS assistance systems in order to predict arrival times. Artificial Neural Network (ANN), Support Vector Regression (SVR) and Long-Short-Term-Memory (LSTM) will be compared with a historical model to determine which model will provide the best results from the data gathered from Entur, Norway.(*Entur, https://www.entur.org/om-entur/* n.d.). The bus-connections and lines we are using, are from Ruter, who is handling the transportation system in Oslo, Norway. A more detailed description of our assignment is located under Assignment details on page 2.

1.2.1 Research questions

How do artificial neural network models, support vector regression models, long short term memory models, and historic data models compare in predicting the arrival time of public buss transport in an urban area?

Different machine learning models all have their pros and cons and areas where they excel and others where they are lacking. As noted, our endeavor is to investigate different ML models and compare their ability to predict correct arrival times of buses. This will be accomplished by feeding the models identical data and giving them the same tasks, we will be positioned to perform a fair comparison between the models and identify their respective strengths and weaknesses, as well as their ability to solve the task.

Then, it is pertinent to ask "what are the challenges of using machine learning models when predicting arrival time in an urban environment using vehicle monitoring data?"

Most machine learning models have their own quirks and needs to perform optimally. Urban environments also have a certain aspect of unpredictability which can be hard to accurately portray. These obstacles can make it hard for models

to perform optimally and to have the ability to make accurate predictions. To arrive at an informed decision on the model of choice, it is vital to identify challenges the different models face in solving this problem.

1.3 Organization of the Thesis

In section 2 (Theory), we review previous studies implementing some or all of these models. To widen the scope, we also provide details on some additional models that can inform inquiry on optimizing arrival time predictions. This information form the basis from which we will forward a hypothesis regarding the optimal estimation model. In the methods section we provide in-depth information regarding where we acquired the data needed for our models, and how we processed our data in order for our models to be able to use it for their calculations. In this section we also describe how we implemented our models. In section 4 (Results), we will present our results accumulated from the models emphasizing the comparison between the models. We will discuss our results in section 5 (Discussion). We conclude and provide suggestions for future work in section 6. Finally, in Appendix A, a list with acronyms used throughout this thesis can be found.

2 Theory

For this study we implement different Machine learning models to predict the arrival-time for any given public transport agency that has their real time vehicle monitoring accessible through Entur.

Finding arrival times using machine learning or other types of models is an interesting task due to the stochastic behaviour of vehicles in urban areas and more stationary behaviour in rural areas. This is one of the reasons why there is a lot of former studies done on this type of problem. In this study we will focus on the models we see fit best for our problem based upon results accumulated by other research groups as well as further research machine learning models that is more uncommon for predicting arrival times of buses, indicated by fewer research papers. Hence, our study is heavily influenced by previous studies.

2.1 Previous studies

Considerable work has been done on predicting travel time using different types of models and algorithms, such as historic data models, regression models, SVM/SVR, LSTM and ANN. Due to the complexity of the problem with traffic congestion, incidents and dwell times in urban areas and the stationary scope in rural areas, each model tends to imply different advantages and disadvantages. How suited a model is for a certain problem depends on whether the solution needs to be cost effective or accurate.

2.1.1 Historic data

Historic data models use acquired data from previous trips to predict current and future bus travel times. William and Hoel describe this model and conclude that this type of model require a stationary and stable bus environment in order to predict with high accuracy. For this type of model to perform with reasonable values, the wanted location for the data should be placed in a rural environment(Williams and Hoel 2003). A historic data model is often used by researchers and developers for comparison to their own models as it is easy to implement and quick to run (Jeong and Rilett 2004)(Altinkaya and Zontul 2013)(Petersen, Rodrigues, and Pereira 2019)(Zou, J. Wang, and Chang 2008).

2.1.2 Regression model

Regression models unlike historic data models, do not depend on the data to be in a set stable condition (Altinkaya and Zontul 2013). A regression model uses independent variables and measures their effect simultaneously. (Amita, Singh, and Kumar 2015). The studies from Rilett and Patmaol both used regression models, either for comparison to other models, or as their own implementation to the problem. However, both studies found ANN to be the superior model to predict with higher accuracy (Patnaik, Steven I-Jy. Chien, and Bladikas 2004)(Jeong and Rilett 2004). Nonetheless, an advantage of using regression model is the models' ability to find factors with high impact to traffic flow and delays or factors with less importance. For instance a study conducted by Patnaik et al. found weather data to be less of an importance to optimize the model(Patnaik, Steven I-Jy. Chien, and Bladikas 2004). However, the determining factors are heavily influenced by the geo-location of the data. To illustrate, in Norway, where the weather may change rapidly, a sudden snowfall may decrease the traffic flow, and hence be of substantial importance to predicting arrival times, but the accuracy of these predictions is reduced due to the comparative unpredictability of snowstorms. By contrast, in countries where there is typically slower and less pronounced changes of weather (e.g. Greece), weather data can explain less of the variance in travel time deviations, even though with higher accuracy. Hence, the generalizability of regression models regarding the variables needed in the model as well as the strengths of the regression coefficients may be limited due to high variability between locales.

2.1.3 Support Vector Machine / Support Vector Regression

Support Vector Machine and Support vector Regression Model (SVR) are not that common to use to predict arrival times compared to ANN and historic models, as indicated by the comparative infrequency of research papers and former studies using these types of models to predict travel times. Even so, some have compared SVM and SVR with ANN or other types of models(Altinkaya and Zontul 2013) (Bin, Zhongzhen, and Baozhen 2006) (Wu et al. 2003). Conceivably, vector based models can behave extraordinary well with time series due to the models' ability to generate a unique global solution for any given training data. Such a contention concurs with the results from a study conducted by Chun-Hsin Wu et al. where they found SVR and SVM too both outperform their compared historical models in finding an estimated travel time (Wu et al. 2003). A sizeable advantages of using SVM with datasets that require a considerable amount of training, is the way SVM handles overfitting. If the regularization parameters are controlled correctly, overfitting should not be a problem.(Olson and Delen 2008) However, a study conducted by Bin, Zhongzhen and Baozhen found that the performance of the models gradually decreased when the size of the data increased. This is due to the large time of computation needed.

Even so, they found SVM to outperform ANN with their initial dataset (Bin, Zhongzhen, and Baozhen 2006). The datasets need to contain all the historic and real time data for predicting an optimal travel time, such as passenger count using APC, location using AVL, speed of the unit, weather data from a third party organization, traffic data (often from a third party organization) etc. In consequence, datasets will eventually grow substantially, a fact that is hard to minimize. This causes long computations, an inherent problem with SVM and SVR acknowledged for a long time. Thus, in those areas where the datasets become very large, ANN or other models will probably accumulate the results much more efficiently. (Olson and Delen 2008).

2.1.4 Long short term memory

Long short term memory (LSTM) is a model fully capable of computing arrival times from real time data. Hochreiter and Schmidhuber researched this model in 1997 and wrote that LSTM is proven to be robust for capturing long-term dependencies using an input, forget and output gate to maintain the information of a cell as well as a state variable. The LSTM network will then be able to maintain the state of a cell from previous observations, as well as throwing away irrelevant information.(Hochreiter and Schmidhuber 1997) (Petersen, Rodrigues, and Pereira 2019). The latter research group used LSTM to find arrival times in an urban area. Peterson, Rodrigues and Pereira created 2 different LSTM models in their study. One containing convolutional filters in order to find the relationship with buses on cross-linked lines, and one without for comparison (Petersen, Rodrigues, and Pereira 2019). A convolutional neural network is mainly used for image recognition due to its ability to connect patterns from neighboring pixels by using the filters to compute the correlation between the pixels (Simard, Steinkraus, and Platt 2003). The study done by Peterson et al. also made a historical model for comparison to their own model as well as testing it against predictions from Google Traffic. They got their data from Automatic vehicle Location installed on the vehicles. Their results revealed that the LSTM model using convolutional filters outperformed the comparison models. However, the pure LSTM model also outperformed the historical model and google traffic prediction in the early peak hours. Even so, the same model gets beaten by the google prediction when the traffic flow deviates from the normal pattern. The pure LSTM model is created similarly to the study by (Duan, Yisheng, and F. Wang 2016). In their model historic data was used as input to the network to obtain the LSTM calculations to find the desired output. Duan et al. trained their model from data retrieved from Highways England. Even though their results yielded relatively small prediction errors and mean absolute errors, they were not fully comparable to models computing the complex travel time problem regarding urban areas. This is due to the fact that highways contain less traffic congestion caused by intersections and short dwell times.

2.1.5 Artificial Neural Network

Artificial Neural Network (ANN) is the most commonly used machine learning model for predicting arrival times and there exists multiple former studies within this area (Jeong and Rilett 2004)(Agafonov and Yumaganov 2019)(Altinkaya and Zontul 2013)(Amita, Singh, and Kumar 2015)(Steven I-Jy Chien, Ding, and Wei 2002). There are also a substantial amount of research papers conducted prior to the 21th century. However, we will focus on the more recent studies, even though they are all based upon previous work. One of the main reasons for the popularity of ANN is due to its ability to perform with non-linear relationships (Altinkaya and Zontul 2013). Using multiple layers, ANN has a unique way of calculating predictions with high accuracy using weighted variables set by the developer. These weights are set more or less empirically by testing and tweaking the parameters. In a study from Russia by Agafonov and Yumaganov in 2019 they made a basic ANN model with the factors of a few different speed measurements gathered from installed systems on buses to find the travel time. They also made an extended ANN using more speed measurements, such as average and current traffic flow. They described the weather data, occurrence of congestion and incidents to be important factors for calculating travel time, and concluded that they should be accounted for. In their study they assumed anomalies in the traffic-speed measurements would reflect these factors indirectly (Agafonov and Yumaganov 2019).

In a study done by (Jeong and Rilett 2004) they constructed an ANN model, comparing it to a regression model and a historical model. They underscored that dwell time and the consideration of traffic congestion is important factors for causing traffic delays. They used Automatic Vehicle Location (AVL) together with the bus schedule in Houston, Texas in order to obtain the appropriate data. And they used the backpropagation algorithm which is the most common algorithm for transportation problems. The ANN-model made by Rilett and Jeong outperformed the other models considerably, and they hypothesized that the results came from the ANNs' great ability to identify the nonlinear relationships in finding travel times (Jeong and Rilett 2004). However, Chien et al. pointed out in their study that the backpropagation algorithm has a lengthy learning process and might be hard to apply to an online application (Steven I-Jy Chien, Ding, and Wei 2002)

In sum the artificial neural network models perform decently, and typically better regarding the problem of calculating travel times compared to other models. This might specially so in urban areas where the solution needs to solve a stochastic problem with dynamically changes to the variables.

2.1.6 Summary

Most of the previous studies we encountered are using ANN as their main model, and found it to outperform other models in urban areas (Altinkaya and Zontul 2013)(Jeong and Rilett 2004)(Agafonov and Yumaganov 2019)(Amita, Singh, and Kumar 2015). Such an overall conclusion does not come without exceptions; Baohzen et al. found SVM and SVR to outperform ANN in their results. However, they also noted that SVM and SVR required higher computation times with bigger datasets. Similar to SVM, predicting arrival times using LSTM as a model has not yet been well documented. As mentioned earlier, the newer study done by Peterson et al. found LSTM to be usable to solving these types of stochastic tasks where their model outperform both their historical model and googles' own prediction system. Nonetheless, Peterson et al. did not compare their model to an Artificial Neural Network model.

3 Method

Given the models' popularity and likely efficiency, we decided to use a simple ANN, advanced ANN, SVR and LSTM and compare these models against a historic data model. Specifically, we included ANN due to the long array of favorable reports of efficiency (albeit with exceptions). SVR was included because it might have considerable potential in solving the regression problem related to the study, even though it has seldom been compared to ANN or LSTM. We included Long-Short Term Memory model for the same reason as SVR. Finally, because Ruter is likely currently using historic data based upon a qualified guess from their own "estimated-arrivaltime", we will compare the above models to such a historic data model.

3.1 Hypothesis

The findings in Section 2.1 led us to hypothesize that an advanced ANN model will have higher accuracy in predicting arrival times than hard-coded algorithms, here represented by average bus time calculations and a simple historical model. The above findings and our theoretical reasoning led us to believe that SVR and LSTM will outperform the historical model. As regards to the artificial models, we believe ANN would be to prefer to SVR and LSTM in any real-world application due to its ability to perform with non-linear relationships and performances with big datasets. However, we believe the machine learning models will predict with approximately similar results, with ANN having slightly better predictions.

3.2 Datasource

For this study, we based our models on two types of datasets. Public transit data acquired from Entur using vehicle monitoring, and weather data acquired from the Norwegian Meteorological Institute.

3.2.1 Public bus data

Entur which is a Norwegian organization for travel-planning that acquires data from 58 different travel agencies, including the transportation agency located in Oslo, Ruter. (*Entur*, https://www.entur.org/om-entur/ n.d.). Entur provided us with two types of data which we could use. One of the datasets included historical data similar to a database table where the buses logged actual arrival time to their appointed bus stop. However, we expected this data to be constructed in a way that would give false positives to our models. This is due

to the fact that every bus that was on time, arrived at the appointed bus stop exactly the same time (on the second) as the expected time, which is highly unlikely in practice. In order to train our models we had to remove this dataset out of our source. The second data provided to us was a vehicle monitoring-data. Here, the buses logged their current location as well as their progress from one bus stop to the next stop with a value measured in percentage. This type of data was acquired every day from 01.11.2019 to 24.02.2020. Which meant we had 115 days of datasets to work with and 39,385,813 data points before sanatization and 33,757,295 after. To be able to use this data, we had to do quite a bit of data processing which will be described in section 3.3

3.2.2 Weather data

The weather changes quite rapidly in Norway, especially during the winter. This fact relates to one of our hypotheses to whether or not this will cause traffic delays. The data we chose to use in our models were gathered from the weather station at Blindern in Oslo. Due to the fact that weather conditions tend to cover a large area we chose to generalize the data gathered here to fit all the edges in Oslo. The data we chose to use is the SYNOP Code, which is a code from 0-99 which indicates the weather conditions at that time. (*SYNOP Codes, https://orap.met.no/Kodeforklaring/Kodebok/koder/ww.html* n.d.). Data was gathered from the Norwegian Meteorological institute. We had data covering almost all the bus data points we had, with a few exceptions during nighttime. We then made a script which iterated through all buss data points and matched them with a corresponding SYNOP code for that time. In the case there was a hole in the weather data and such no corresponding SYNOP code, we would use the code that was last reported. By doing this we successfully managed to add weather condition feature to all our bus data.

3.3 Data processing

As mentioned earlier, we had one type of data which we could use, the one provided with vehicle monitoring. In order to extract the features and labels for our models to train on we had to fetch out and map the timestamp of the buses according to their progress measured in percentage on the route. To get the features for our models we divided all the routes into edges and vertices where each vertice represented a bus stop and the edges represented the segment between each bus stop. This type of segment-splitting was also conducted in the study by Agafonov and Yumaganov in 2019 (Agafonov and Yumaganov 2019). We collected every bus that had a record for being between a set of vertices and tracked their location and timestamp. We then had to fetch out their first instance of arrival to their appointed bus stop in order to get the labels. Too remove any duplicate labels we removed any records that was logged after the

first record for the specific bus. However, if the timestamp for continuously staying at the same bus stop was more than 10 minutes, we assumed the bus had moved on and was possibly on its next round, which then was recorded. The code below shows a snippet of unprocessed data followed by the same processed data.

raw_data.xml :

```
<ResponseTimestamp>2020-04-01T20:02:23.127695+02:00</ResponseTimestamp>
            <VehicleActivity>
                <RecordedAtTime>2020-04-01T20:02:10.774+02:00</RecordedAtTime>
                <ProgressBetweenStops>
                    <LinkDistance>451</LinkDistance>
                    <Percentage>3.55</Percentage>
                </ProgressBetweenStops>
                <MonitoredVehicleJourney>
                    <VehicleRef>103027</VehicleRef>
                    <PreviousCalls>
                        <PreviousCall>
                            <StopPointName>Tjuvholmen</StopPointName>
                        </PreviousCall>
                    </PreviousCalls>
                    <MonitoredCall>
                        <StopPointRef>NSR:Quay:7791</StopPointRef>
                        <VisitNumber>2</VisitNumber>
                        <StopPointName>Observatoriegata</StopPointName>
                        <VehicleAtStop>false</VehicleAtStop>
                        <DestinationDisplay>Helsfyr T</DestinationDisplay>
                    </MonitoredCall>
                    <OnwardCalls>
                        <OnwardCall>
                            <StopPointRef>NSR:Quay:7747</StopPointRef>
                            <VisitNumber>3</VisitNumber>
                            <StopPointName>Lapsetorvet</StopPointName>
                        </OnwardCall>
                    </OnwardCalls>
                </MonitoredVehicleJourney>
            </VehicleActivity>
...
```

processed_data.csv :

```
Date,RecordedAtTime,WeekDay,VehicleRef,LastStop,NextStop,Percentage
0401,200210,2,103027,Tjuvholmen,Observatoriegata,3.55
...
```

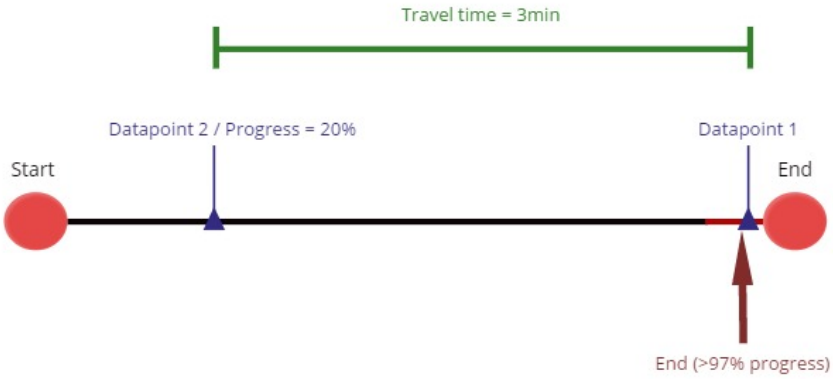
3.3.1 Data extraction method 1

We tested two methods of creating feature and label sets using the data filtered applied from the method mentioned above. An example of the first method we can see in figure 1. This method, as well as the second we will discuss later all rely on using the buses progress report system. As the buses report how far they are on a route in percentages from 0-100% we use this combined with when they reported their progress to estimate how long they used to travel a certain percentage of an edge. To account for some inaccuracies in the progress report we decided to count all reported progresses above 97% as the buss having completed the route and at the buss stop. So the first thing we do is to find a datapoint where the bus reported a progress of above 97%, this bus we count as having completed the edge presented as datapoint 1 in figure 1. We then search through the edge for another datapoint that matches these criteria:

- Must be same edge
  To make sure that we do not get data from a previous edge.
- Must be same vehicle reference
  Makes sure that it is the same buss.
- No more than 10 minutes back in time
  Makes sure that it is the same trip.
- Must have at least 5% less progress
  To make sure that the buss is not standing still and thus a duplicated listing.

Any data listing that matches these criteria will be a bus on the same edge on the same trip but at previous point on the edge, this listing will now be know as data point 2 in figure 1. We now have two data points for the same bus, on the same edge, on the same trip but at different locations on the edge. The next thing we do is to calculate the time difference between data point 1 and 2, more specifically the time it takes for the bus to travel from datapint 2 to datapoint 1. This is simply calculated by taking the difference in when the two points were reported, this difference will be the label of the dataset, this is 3 minutes in figure 1. The features will be the time datapoint 1 was reported, the progress of the bus at datapoint 1, in the case of figure 1 would be 20%, the weather type (SYNOP code 3.2.2) and weekday. The date is also included but only for sorting and reference purposes and is not passed as a feature to the models. If we had comprehensive data spanning multiple years it could be included so the model could learn trends for specific dates.

Figure 1: Data extraction method 1



3.3.2 Data extraction method 2

There is one significant problem with our dataset and the extraction method used above, the dataset we have is so small and the restrictions for extraction so strict that it yields very few usable feature label pairs. By using the method above we were able to only extract 3,915 feature-label pairs from the edge "Nasjonalteateret - Vika atrium" containing 76,870 individual reported datapoints. With this few usable feature-label pairs the accuracy of our models will suffer greatly.

To remedy the problem mentioned above we chose to change the criteria for datapoint 1, the end datapoint. In the first extraction method datapoint 1 has to be in the end zone, meaning it has to have a progress of above 97%, in the second extraction method there is no such criteria. This means that any datapoint can be an end point no matter where they are on the edge, they only have to have a corresponding point that was reported earlier on the edge, the criteria for datapoint 2 remains the same as described in the method above. See figure 2 for an example of this. After we have found datapoint 1 and 2 we now set the progress of datapoint 2 to be its original progress and how much progress datapoint 1 has left before it reaches 100% added together. We are effectively moving the two datapoints further on the edge so datapoint 1 is at the end of the edge and the progress of datapoint 2 is scaled according to how far they had to be moved for datapoint 1 to be at 100%. We can see the original location of the datapoints on the edge in figure 2 and their locations after it is moved in figure 3. In figure 2 datapoint 1 is at 60% and datapoint 2 is at

20%. In figure 3 we have moved both of the points 40% further along the edge, datapoint 1 now has a progress of 100% and datapoint 2 has a progress of 60%. The progression the bus has to make from datapoint 2 to datapoint 1 remains the same. The calculation of the time the bus takes to travel the given progress is calculated the same way as before. By using this method of extraction we are able to gather much more data, from the 76,870 individual reported datapoints on the edge "Nasjonalteateret - Vika atrium" we were able to extract 167,702 feature-label pairs, a significant increase from the first method.

Figure 2: Data extraction method 2 part 1

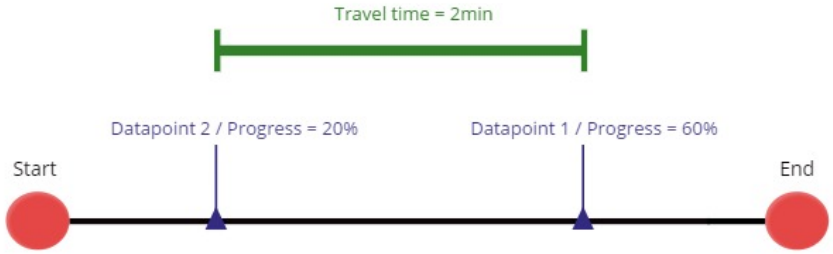
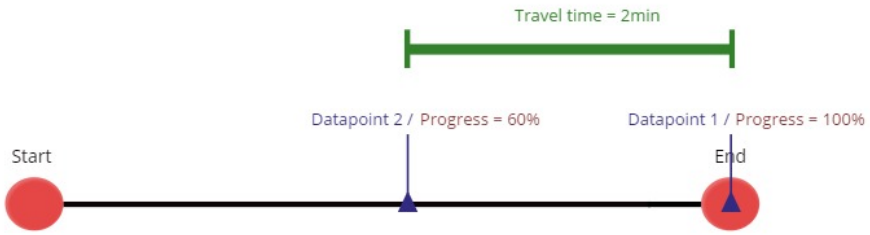


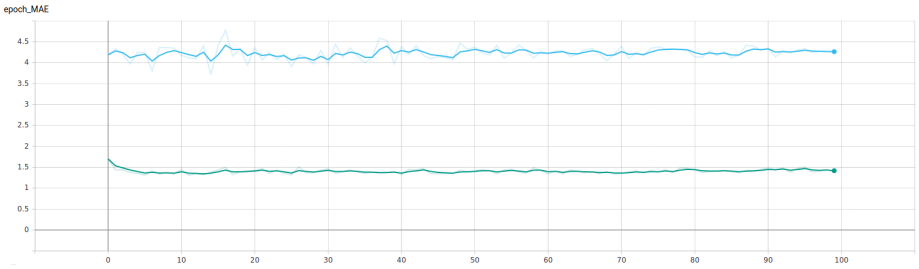
Figure 3: Data extraction method 2 part 2



3.3.3 Comparison of extraction methods

To choose which of the methods we would use, they were both tested to see which of them gave the best results. To accomplish this, they were both trained and tested on the same Advanced Neural Network with the same attributes. The results of this is illustrated in the graphs presented below in Figure 4, which illustrates the Mean Absolute Error of each data set. The blue line (top) representing the first method with only labels above 97% and the green (bottom) the other method where labels can be any point with a given earlier datapoint.

Figure 4: MAE short vs long



As we can see, the augmented dataset performed substantially better with an MAE of around 1,4 vs 4,3 for the original non-augmented dataset. The augmented dataset did take longer to train, with 100 epochs taking 25 minutes while the small dataset only took one minute for 100 epochs. Based on these findings we decided to continue with using the augmented dataset created by using the second data extraction method(3.3.2).

We are aware that by using the second extraction method 3.3.2 we could be introducing inaccuracies if it takes longer to travel one part of the edge than another. If for example the first 50% of the edge takes a longer time to travel than the last 50% and we move a datapoint pair that was originally between 10% and 40% to 70% and 100%, then the time it takes to travel the last 30% of the edge will not be the true time it takes as it is not actually the last 30%. But knowing this we believe the difference in time it takes to travel will not vary too much across one edge as they are usually very short, and that having data points from multiple places on the edge will average this out and minimize the total error. All models will be fed identical dataset, such as no model will have more accurate data than the other.

3.3.4 Encoding time feature

Another portion of the data we had to manipulate were the features that represented the time of day the buses reported where they were. Due to time being cyclic with seconds and minutes being base 60 and hour being base 24 the model might encounter a problem understanding the actual difference between two timestamps. An example of this is minute 59 at 11 o'clock and minute 01 at 12 o'clock, due to the cyclic nature of time these two timestamps are only two minutes apart. A machine learning model based on the base 10 numerical system will more likely see them as being a whole 58 minutes apart. This could create a problem with it learning the relations of two close datapoints which it interprets as being far from each other.

To solve this problem, we chose to encode the hours, minutes and seconds using the sine and cosine trigonometric functions. Due to these functions cyclic nature they will allow us to correctly represent the cyclic nature of time and the true distance between two times even after they have clocked over. As both the sine and cosine functions can map to the same values given different inputs, we had to use both functions, as they will overlap to one unique value.

These are the functions we used to calculate the sine and cosine encoded values. The variable data will be the original value we wished to encode and the data_base represents the original base for the number we are encoding, here 24 for hour and 60 for minute and seconds.

$$data\_sin = \sin(\frac{2*\pi*data}{data\_base})$$

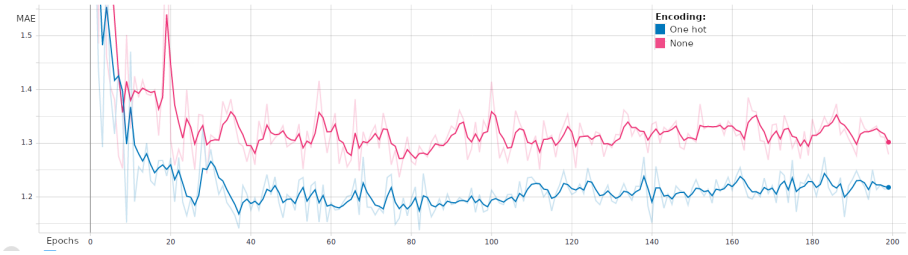
$$data\_cos = \cos(\frac{2*\pi*data}{data\_base})$$

The resulting sine and cosine pairs could now be used in the dataset in place of the old hour, minute and second data.

3.3.5 Encoding weekday feature

Another cyclic categorical feature in our out-data are weekdays. As traffic will be different on a Sunday than on a Wednesday, they are an important factor to include. Although weekdays are cyclic, we do not wish to represent them as such as we do not intend the model to think that Monday and Tuesday are inherently more alike than Monday and Thursday due to their proximity. To solve this issue, we encoded the weekdays in a one-hot vector as to represent all the weekdays in a numerical, normalized way so they do not introduce any bias to the model simply based on their proximity to each other. Encoding them as a one hot also avoids weekday 6 being weighted more than weekday 0 due to it being a bigger number.

Figure 5: MAE weekday encoding difference



The benefit of encoding the weekday as a one-hot vector vs. encoding it as a sine cosine pair can be seen above. In the figures, we see that the dataset with the one-hot encoded weekday features (blue) had a lower Mean Absolute Error (figure 5) compared to the dataset with the cyclic sine and cosine encoded weekday feature (pink).

3.3.6 Data after preprocessing

In Table 1 presented below, we see how an instance of a feature-label pair will look in the final dataset. With this dataset we hope to be able to train a model who can predict how long it will take to travel the rest of an edge given how far a bus is currently on the edge.

Table 1: Final features and label

Feature	Label
Date Percentage WeatherType Feature_hour_sin Feature_hour_cos Feature_minute_sin Feature_minute_cos Feature_second_sin Feature_second_cos weekday_0 weekday_1 weekday_2 weekday_3 weekday_4 weekday_5 weekday_6	TimeToTravel

3.4 Training

With the data processing described in section 3.3, we had two methods for training our models. The first method which we called "edge based training", had more data due to the fact that we could train on multiple lines as long as they were driving on the same edge and stopped at the same connected bus stops. This "edge based training" also implied that we were able to utilize all of the identified feature-lable pairs, where this type of training were not dependent on the entire route. We chose the nodes "Nasjonalteateret" and "Vika atrium" with its corresponding edge for training our models. The reason being the amount of feature-lable pairs was a lot higher on this edge than the majority of the other edges. At the same time, this edge included traffic obstacles, such as roundabouts and junctions. In the figure below (Figure 6), we present the edge "Nasjonalteateret-Vika Atirum" as node 1 and 2. The lines 81, 32 and 33 are the bus-lines that all has "Nasjonalteateret" and "Vika Atrium" as connected bus stops. All of these bus-lines will provide us with valuable datapoints, even though their final destination are different from one another.

Figure 6: Edgebased

Another method for training our models was focused on each complete route. In order to do so, we had to fetch out each bus-line and train our models for each iteration of the route.

Figure 7: Linebased

This method came with a data issue. As the historical data given to us from *Entur* had major faults, and we were using vehicle-monitoring data. The issue here was quite simply lack of data. For this method to work, we needed the current line we were training on to have logged data from each edge at every full iteration. If not, we were missing an edge, and the route would be incomplete. This would cause our models to behave abnormal and insufficient to our needs. Due to this, we used the edge-based method to train our models.

3.5 Optimization

When developing a machine learning model you often do not know the optimal parameters. The estimate can be derived from former studies, other sources, and experience with machine learning. But as stated by (Fabrikant 2019), and echoing our critique of the regression approach, the parameters can vary from place to place and city to city. In lack of such comparable data, and wanting to find the best parameters to use for our particular model, dataset and application, we therefore did tests and gathered empirical analytics to base our research on.

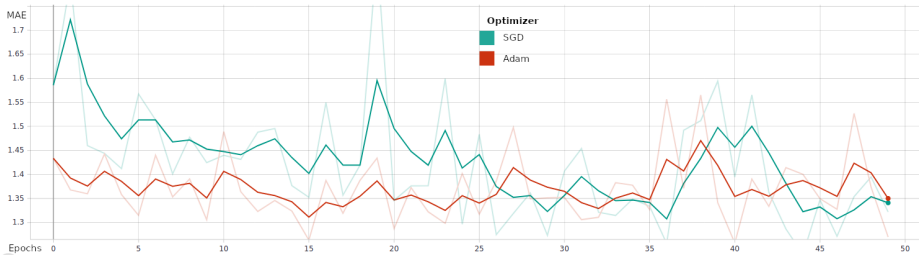
The main areas where we could optimize our models were: Using different optimizers such as Stochastic Gradient Descent (SGD) and the Adam algorithm, utilizing different learning rates, using different losses like Mean Absolute Error (MAE) and Mean Squared Error (MSE).

An important problem we had in mind when designing our models, was the overfitting problem. As we did not have an enormous amount of data, overfitting our models is a problem that could realistically happen. To prevent this, we implemented two methods known for solving this issue, the use of dropout and splitting our data into training, validation and test sets. For the dropout we chose a rate of 20%, this should help the network be properly regularized, lowering the variability of the network and preventing overfitting. As the next measure to prevent overfitting we chose to implement, is to use train, validation and test datasets. The train/validation/test split we chose was 70%/20%/10%, this provides the networks sample data to train on, and we still have enough data for accurate validations and tests. By implementing these two methods we believe our results were relatively safe from being invalidated due to overfitting.

3.5.1 Optimizers

The two optimizers we encountered the most in reading research papers were the Adam (Kingma and Ba 2015) and Stochastic Gradient Descent (SGD) (Genevay et al. 2016). As these were the ones we encountered the most and seemed to be the possible best fits for our problem, we decided to test the two optimizers and see which preformed the best. We ran a simple neural network with the same hyperparameters individually, first with the Adam optimizer and secondly with the SGD optimizer. As seen in the figure below (Figure 8) the Adam optimizer (brown) preformed notably better then the SGD optimizer (green), albeit SGD was 8.2% faster than the Adam optimizer. As Adam provided a marked improvement in MAE over the SGD optimizer, we chose to use Adam for our models.

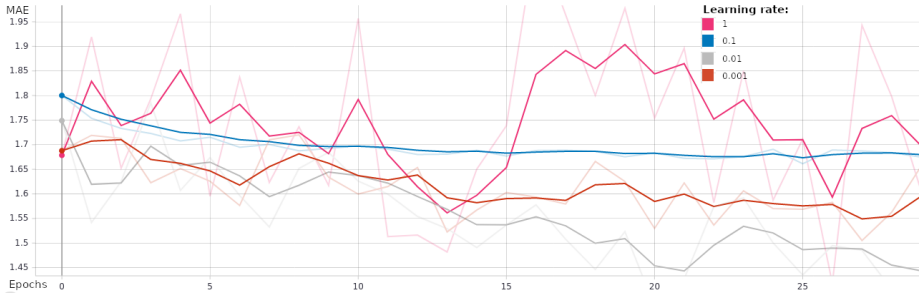
Figure 8: MAE of Adam and SGD optmiziers



3.5.2 Learning rates

To find the optimal learning rate for the simple neural network, we tested varying learning rates empirically to see which was the best, the MAE of each is illustrated in the graph below (Figure 9). The orange represents a learning rate of 0.001m, the gray a learning rate of 0.01, the blue a learning rate of 0.1 and the pink a learning rate of 1. As we see, the optimal learning rate was 0.01. With a learning rate of 1 it became too unstable and jumped around while a learning rate of 0.1 was more stable but had worse MAE. The learning rates of 0.001 and 0.01 had both good MAE, but out of the two the optimal one was the 0.01, which is the one we chose to continue with.

Figure 9: Loss at different learning rates



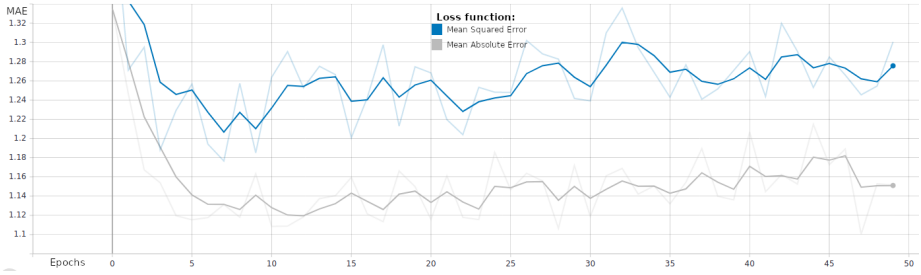
We repeated this process to find the optimal learning rate for the Advanced Neural Network as the two models do not necessarily have the same optimal learning rate. Although that being the case, in this instance the optimal learning rate proved to match the model above, with the most optimal being a learning rate of 0.01, followed by 0.001 and the least optimal being a learning rate of 1.

3.5.3 Losses

To find the best loss function for our problem we decided to test between the two most common loss functions used for regression problems, those being Mean Absolute Error (MAE) and Mean Squared Error (MSE). The MAE loss function is less sensitive to outliers while the MSE function is more sensitive, due to it using the squared of the error and such outliers will have a bigger impact on the error than just taking the Mean Absolute Error. From testing both of these loss functions on the same model, we see that the MAE loss function performs with a notably better accuracy than the MSE function. This is illustrated in the graph below (Figure 10) comparing the MAE of the MAE (gray) and MSE(blue) functions. The results were the same when MSE was used as a

metric for measuring which was best. This is possibly due to the dataset having a non-negligible number of outliers in it. As such, we decided to use Mean Absolute Error in our regression models as it proved to be the best fit for our dataset.

Figure 10: MAE with MAE and MSE as loss function

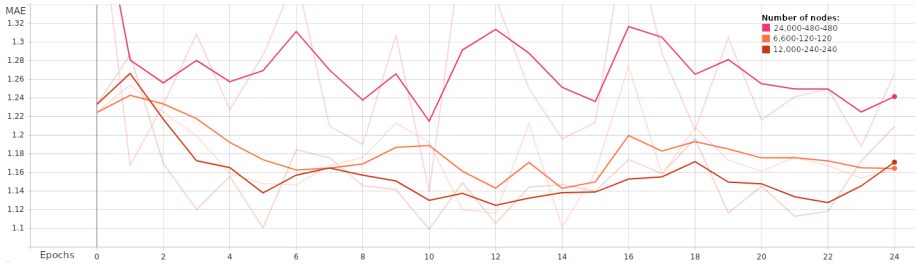


3.5.4 Size of layers

Next, we aimed to find what number of nodes would be optimal for the simple neural network. For this, we ran the model with one hidden layer and the same learning rate of 0.01 but with a varying number of nodes in that layer. The number of nodes in the layer we tested were 12,24,48, and 96 nodes. From this, the conclusion was that with 96 nodes the layer preformed the best. In all, the layer with 12 nodes preformed the worst. Even so, it should be noted that from worst to best there was a very small difference. Thus, the layer of 96 nodes was the one we chose to continue with, although the layer with 48 nodes was slightly quicker to train and provided quite similar results.

For the Advanced Neural Network, we aimed to test a network with 3 hidden layers. We tested one input layer with a high number of nodes and two having a smaller number of nodes to see if the increased hidden layers and nodes made the model better at separating the features and generalizing the data. We chose to test three different models, all with three hidden layers. The first we tested contained 6,600 nodes in the first layer and 120 in each of the following. The second had 12,000 nodes in the first layer and 240 in the two following. The final model had 24,000 in the first, and 480 in each of the following. All models had a learning rate of 0.01. As we can see from the graph below (Figure 11), the model with 12,000-240-240 (dark brown) preformed the best, followed by the simplest with 6,600-120-120 nodes (orange). The worst model proved to be the one with the highest amount of nodes, 24,000-480-480, both in accuracy and time to train, with it taking 53m to train compared to 13m for the 12,000-240-240 model and 8m for the simplest one. As such the model with 12,000-240-240 nodes will be the one we choose to continue with.

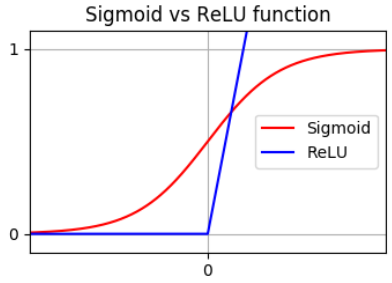
Figure 11: Mean Absolute Error of ANN model with 3 hidden layers given different number of nodes in each



3.5.5 Activations

There are a couple of activation functions we could use, but the most used ones are the Sigmoid function and the Rectified Linear Unit. The Sigmoid function, as you can see in Figure 12 (red), will always return a number but it will never be quite 0 or 1, while the ReLU (blue) takes the $max(0, a)$ and returns 0 until $x > 0$.

Figure 12: Sigmoid vs ReLU function



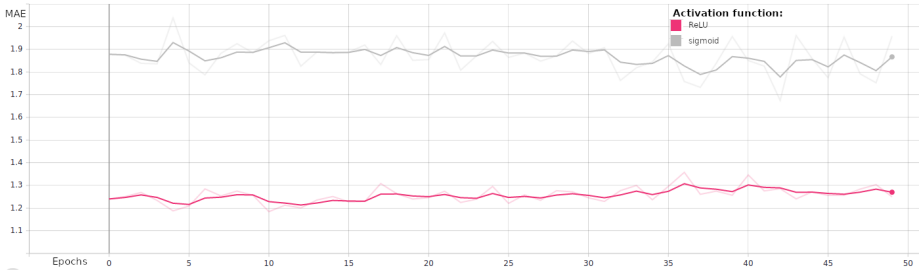
According to (Li and Sanderson 2017); Early networks used Sigmoid to squish the relevant weighted sum into that interval between 0 and 1, motivated by the biological analogy of neurons either being active or inactive, ReLU fits this analogy better. However relatively few modern networks use Sigmoid anymore as ReLU seems to be much easier to train.

The benefits ReLU has over Sigmoid are a reduced likelihood of vanishing gradient and sparsity. When $a > 0$, the gradient has a constant value in contrast to the gradient given by the Sigmoid function. The gradient gets smaller the higher the absolute value of x gets. The constant gradient from ReLU results in faster learning; another benefit is ReLUs sparsity. This happens when $a <= 0$. When more units are less than 0, the sparser the resulting representation gets. Since Sigmoid always returns a value greater than 0 we get a dense representation. Sparse seems more beneficial than dense (Maker 2014).

To verify that ReLU was indeed the best activation for models, we chose to do a simple test where we tested an ANN with the ReLU and Sigmoid activation.

As we can see illustrated in Figure 13 ReLU(pink) preformed the best with the pure Sigmoid model (gray) preforming the worst.

Figure 13: Mean Absolute Error of ANN model with 3 hidden layers given different activation functions



3.5.6 Support Vector Regression specific parameters

For the Support Vector Regression there were two parameters we had to find, the C value and the epsilon value. The C value tells the model how much we wish to avoid miss-classification, the margin of the hyper planes. The epsilon controls how wide the ϵ-zone is, used for fitting the training data. For the C-value we tested 3 values, 0.5, 0.2, and 0.1, from these 3 tests we found that a C-value of 0.2 gave the most accurate classifications. The same method was applied to the epsilon parameter, with the values 0.25, 0.18 and 0.1, with 0.18 giving us the best results. The Radial Basis Kernel was also chosen as our kernel as it is generally the most accurate of the kernels with a tradeoff for time it takes to fit the model.

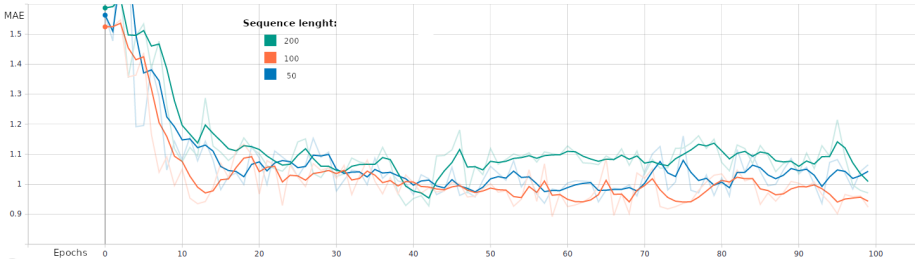
3.5.7 Long-Short Term Memory specific parameters

The Long-Short Term Memory model shares many of the same features as the Advanced Neural Network, as such due to time constraints we chose to use the same loss function that we had found to work best. We also chose to test three different LSTM model of different sizes, one small, one medium and one large. The specifics of these can be seen in the end of this chapter where all the models are summarized 3.6.4. The results of this sections were achieved using the medium LSTM model containing 2 LSTM layers and one dense layer with respectively 64-64-16 nodes.

The notable parameter that we had not seen before was the time step parameter, how long the model will remember earlier values. As before to find this we tested the same LSTM model with multiple different sequence lengths (time steps), this

is illustrated in the graph below (14) From this we found that a sequence length of 100 (orange) to be the optimal value for accuracy. Close behind, we found that a sequence length of 50 (blue) achieves approximately the same accuracy and a sequence length of 200 (green) is the least accurate of the three. Halving the sequence length will also halve the training time, meaning the model with a sequence length of 50 took half the time to train of the model with a sequence length of 100. This is a fact that it is worth taking note of and considering, especially if time to train is an important factor. As we wished to optimize our accuracy we chose to use a sequence length of 100.

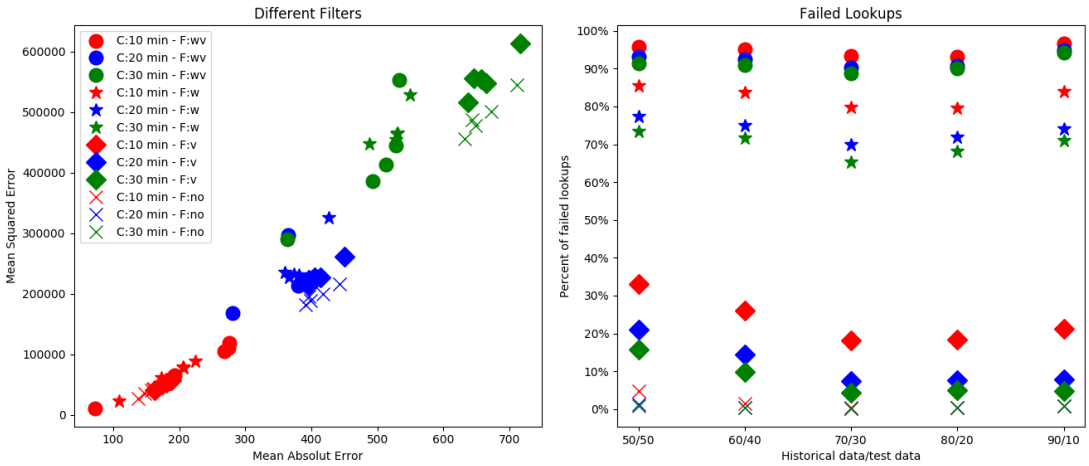
Figure 14: Mean Absolute Error of LSTM model with 3 different sequence lengths



3.5.8 Historical model specific

For the historical model we had to consider the possibility that a historical lookup failed due to lack of data. Given the same dataset as the other models we tried to cut the data on different lookup data to test data ratios, filtered on weekday, vehicle reference, both, and no filter.

Figure 15: Analytic graph of historic model parameters



As shown in Figure 15 on the left, we see that the runs with cutoff time of 10 minutes was substantially better than 20, and 30 minutes. However, on failed lookups (right), we see that with the circle and star which represent the runs with both weekday and vehicle reference filter failed the most with around 90% fail rate. The red diamonds, which represent only vehicle reference filter, and crosses, representing only weekdays filter did much better. The graph in the right has a trend, most notably in the red diamonds, of having a higher hit rate the more historical data it gets to look through. This was not the case with too high historical-to-test data ratio.

When we run this model with the same parameters given to the neural networks with no filters, 80/20 data ratio and 10 min cutoff time, we got red crosses, and the hit rate on 80/20 for red crosses is the same as green crosses and an acceptable level of failed lookups.

3.6 Summary of Models

Below are tables of our parameters used for each of our final models. Their values are based upon tweaking, testing and former studies as described earlier in this section. These machine learning models form the results given in section 4.

3.6.1 Simple Neural Network (SNN)

Our first model will be a simple neural network. It will consist of one hidden layer with 96 nodes, this layer will have the ReLU activation function and a dropout of 20%. It will have a learning rate of 0.01 and use the Adam optimizer.

3.6.2 Advanced Neural Network (ANN)

The next model will be the advanced neural network. This model will have 3 hidden layers, the first layer will have 12,000 nodes, and the two following layers will have 240 nodes in each. All of the layers will have a dropout of 20%, the ReLU activation function and the model will have the Adam optimizer.

Table 2: Neural Network

Parameters	Simple	Advanced
Optimizer	Adam	Adam
Activation	ReLU	ReLU
Hidden layers	1	3
Nodes	96	12,000-240-240
Learning rate	0.01	0.01
Dropout	20%	20%
Epochs	100	100

3.6.3 Support Vector Regression (SVR)

The Support Vector Regression model will be using the radial basis kernel. It will have a C-value of 0.2 and the epsilon parameter will be 0.18.

Table 3: Support Vector Regression

Parameters	Values
C-value	0.2
Epsilon	0.18
Kernel	Radial Basis

3.6.4 Long-Short Term Memory (LSTM)

We settled on testing three final LSTM model, one small, one medium and one large.

Small
The first, simpler model contained two hidden layers, the first layers was a LSTM layer with 32 nodes and the second a dense layer containing 8 nodes with the ReLU activation function, both layers have a dropout of 20%, the model will have the Adam optimizer.

Medium
The second model contained three hidden layers, two of these layers were LSTM layers with 64 nodes in each and a final dense layer of 16 nodes, as before the last layers has the ReLU activation, all layers have a dropouts of 20%, the optimizer is also Adam.

Large
The last model is the large version of the LSTM model. It has three LSTM layers containing 128 nodes in each and then a final dense layer with 32 nodes in it. The layers have a dropout of 20% and ReLU is used as the activation function on the last dense layer, this also had the Adam optimizer.

Table 4: Long-Short Term Memory

Parameters	Small model	Medium model	Large model
Optimizer	Adam	Adam	Adam
Activation dense layer	ReLU	ReLU	RelU
LSTM layers (hidden)	1	2	3
Dense layers (hidden)	1	1	1
Nodes	32-8	64-64-16	128-128-128-32
Dropout	20%	20%	20%
Epochs	100	100	100

3.6.5 Historical

The historical model had no specific filter for day of vehicle reference. It used a data ratio of 80/20 and the search time had a cutoff of 10 minutes.

Table 5: Historical Model

Parameters	Values
Filters	none
Dataratio	80/20
Cutoff time	10 min

4 Results

Using the methods and data outlined in the previous segments we continued to the actual refining and training of the models.

4.1 Performance Comparison

In Figure 16 we present two graphs comparing the performance of the models, the graph on the left illustrates MAE accumulated from our models with exception of the historical model and the graph on the right illustrates the MSE. In Table 6 below, we present the same results with numerical values, including the time it took to train the models. The results from the historical model are also presented in the table.

Figure 16: Mean Absolute Error (MAE) and Mean Square Error (MSE)

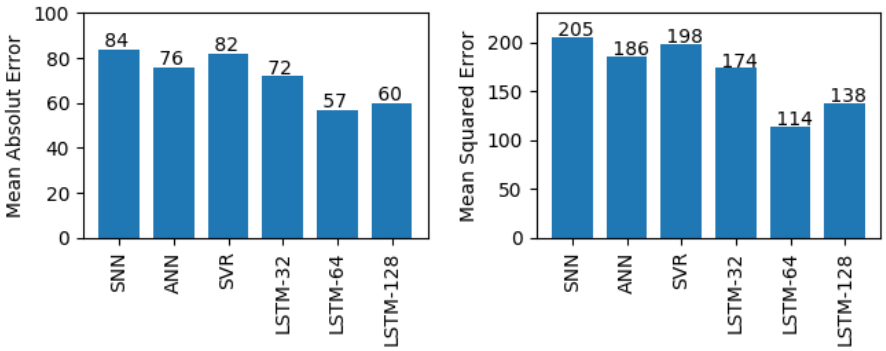


Table 6: Performance Comparison

Model	MAE(s)	MSE(s)	Time to Train
SNN-96	84	205	10m 36s
ANN-12,000-240-240	76	186	1h
SVR	82	198	23h 52m
LSTM-32-8	72	174	4h 17min
LSTM-64-64-16	57	114	12h 6min
LSTM-128-128-128-32	60	138	1d 8h
Historical	148	35,297	1m 13s

4.2 Simple Neural Network

The first model tested was a simple neural network containing one hidden layer with 96 nodes. This simple model has the benefit of being fast to train due to the few nodes and only a single layer. The specific model with 1 hidden layer and 96 nodes took only 10 minutes and 36 seconds to train for 100 epochs, which is fast compared to some of the other models we investigated. This model did as expected compare to the other models, with an MAE of 84 seconds after the 100 epochs trained on and an MSE of 205s.

4.3 Advanced Neural Network

The next model tested was an advanced neural network. This more advanced neutral network consists of 3 hidden layers, the first one with 12,000 nodes, then a second and third both containing 240 nodes. Since this is a bigger and more advanced model, the time it took to train it increased, with it taking 1 hour to complete 100 epochs. This model managed an MAE of 76 and an MSE of 186s.

4.4 Support Vector Regression

The next method implemented was the Support Vector Regression, the regression implementation of the Support Vector Machine classification model. This regressor did comparably well to the Advaned Neural Network, as we can see in Figure 16, with an MAE of 82s and MSE of 198s. The downside to this was the SVR model took a lot longer to train at 23 hours and 52 minutes.

4.5 Long-Short Term Memory

The final type of network tested was the Long-Short Term Memory Recurring Neural Network. Due to its ability to remember previous input patterns and use those in making further predictions proved to be a big benefit in predicting bus arrival times, as we see in Figure 16 it has a lower MSE and MAE than any of the other models.

Small

The small LSTM model was quicker to train than the two other, but in turn had less of an accuracy than the medium and large versions. By running this model for 100 epochs it attained an MAE of 72 and an MSE of 174, beating the previously tested models.

Medium

The medium LSTM model was the most accurate of all the model we tested, as we can see in Figure 16. Although it was not as fast as the small version it still beat both the large LSTM model and the SVR in time to train. This model achieved an MAE of 57s and MSE of 114s after 100 epochs. The model took 12 hours and 6 minutes to complete the 100 epochs of training.

Large

The large model was only slightly more inaccurate compared to the medium LSTM with a MAE of 60s but had even worse MSE of 138s. Notably it took longer to train, finishing 100 epochs in 1 day and 8 hours, the slowest of all the models.

4.6 Historical model

This model suffers greatly from the small datasize, but after analyzing the output from several runs described in section 3.5.8 the result was acceptable. This configuration had a 0.38% fail rate. Time to train in Table 6 presented above, references how long it takes to go through the test data and get the MAE and MSE. These findings are not represented in Figure 16.

5 Discussion

The results accumulated from our models all derive from an identical dataset. This makes for a relatively fair comparison between them, increasing the validity of our conclusion. Even so, our conclusion does not come without limitations and issues which need to be discussed. In this section we will discuss the results accumulated by each of our investigated models, and challenges we faced along our research. Finally, we will discuss the validity and reliability of our research.

5.1 Performance comparison

5.1.1 Artificial neural network

Of all the machine learning models we investigated, the simple neural network performed the worst. It had a Mean Average Error of 84 seconds, meaning in its predictions, the model was off by an average of 84 seconds. This result was expected as it is a simple model with few neurons and hidden layers, which will likely create problems with extracting and conforming to all the features in the data. When experimenting with various numbers of neurons, we observed that higher number of neurons provided better accuracy.

Next, we investigated the advanced neural network. This model consisted of 3 hidden layers, containing 12,000, 240 and 240 neurons in each layer respectively. The model had a notable increase in accuracy compared to the simple model, with an MAE missing 76 seconds on average. The comparatively lesser MAE of the advanced ANN is likely due to its ability to extract more features, and being able to fit the nuances from the data. Although an increase in layers and neurons gave us more accuracy compared to the simple neural network, there was a limit to where the network expansion no longer yielded any benefits and instead decreased the models' accuracy. When investigating the optimization with further increasing the size of the network, we observed that a big network will become overfitted to the data, and the model will then lose some of its ability to generalize. Thus, we believe our final ANN consisting of 3 layers with 12,000, 240 and 240 neurons achieved better results than the simple model, also considering its well balanced ability to generalize the data and to extract features from the data without overfitting.

One positive aspect of the ANN-models compared to the other models we tested, is the fact that they are relatively simple to train. The time it took to train these models is counted in minutes, whereas the other models took hours and even days to complete in some cases. As such, it is an easy model to apply and experiment with as one can easily check how the model performs, thus, tweak it accordingly.

These performance outcomes of both the simple and advanced ANN, did not concur with our hypothesis. Although they both performed better than the historical model we created, their performance did not compare well in most respects to the other models, with the simple network being the worst and the advanced ANN only being marginally better than the SVR-model. The reasons for this could be many, one of being that we did not have perfect data to accurately represented all the factors that can cause delays and decide the travel times. Challenges regarding our datasets are detailed in Section 5.2.1.

5.1.2 Support Vector Regression

The Support Vector Regression model performed mostly as we predicted based on the information obtained from our review of the literature. As we presented in the theory section, using larger datasets makes the SVR-model less effective and likely outperformed by ANN. As we see in the results, the SVR performed slightly worse than the ANN model with an MAE of 82 seconds, albeit it slightly outperformed the simple neural network. One of the reasons for this could be that SVM models that the SVR model is derived from are not good at differentiating between false and positive features, giving equal weight to both useful and noisy aspects. Given that our dataset likely contained some potential uncertainties and deficiencies, there could have been some noise introduced to the model that negatively affected its performance. In addition, as stated in the theory section, the model proved very costly to fit. The model took almost one day to complete its fitting, whereas the advanced ANN accumulated better results and only spent 38 minutes training. From our results, we conclude that using an SVR-model provided no added accuracy or effectiveness compared to applying ANN.

5.1.3 Long Short Term Memory

Of the models we investigated, the Long-Short Term Memory model proved to be by far the most accurate of the models, especially the more advanced ones. The simplest of the LSTM models did comparably well in accuracy terms to the advanced ANN model, outperforming it by a mere few seconds in the MAE metric. However, the two other more advanced LSTM models did substantially better than all the others, with the medium achieving an MAE of 57s and the large an MAE of 60s, meaning the medium achieved an MAE of 15s better than the simple LSTM model and a whole 19s better than best non-LSTM model, the ANN. The probable reason for this superiority being that the memory of the LSTM contributes to the models' knowledge concerning the context of the environment. For instance, there could be a substantial amount of instances that are not clearly represented in our features, but can be induced from the context of a time series, such as traffic or similar conditions. The models'

ability to remember the internal state that contextualizes information is possibly providing it the upper hand in comparison to the other models.

The fact that the simplest LSTM model did not perform as well as the other LSTM models might be attributed to the same problem that the simple neural network model we tested suffered from. The fact that it is constructed with less layers and neurons makes it hard for the model to extract features from the data. If the model suffers from poor feature-extraction and becomes underfitted, it will lead to less received contextual information and unfavorable generalization of the data.

One thing that should be noted regarding our LSTM models is the fact that they take a long time to train compared to the simple and advanced neural networks, especially the more advanced LSTM models. This is one of the reasons one has to properly optimize and evaluate the need for accuracy vs time to train. The medium LSTM-model had a 19 second decrease in MAE over the advanced ANN model, but as a tradeoff it took 12 times longer to train. If one has to maximize accuracy, then LSTM seems to be the clear way to go. However, if the time to train the model also is an important factor, then one might be better off looking at some of the less costly neural networks.

5.1.4 Historical model

Finally, we constructed the historical model for comparison to our machine learning models. While it is hard to outperform the very short time it takes to *train* such a model, this advantage must be balanced against its accuracy. The historical model produced the worst results by a large margin, having almost double the MAE compared to the worst machine learning model. On an edge that usually only takes few minutes to drive, this model has an MAE of 148 seconds and an MSE of over 35,000 seconds. From our research of previous studies, the historical model has been proven to be unfit for predicting arrival times in an urban environment.(Williams and Hoel 2003). From our own study with the results accumulated by the historical model, we can approve of their conclusion.

Although the lack of data might have been the main cause of the bad predictions accumulated by the historical model, one must keep in mind that the other models had the same data to work with. When we compare the results from the simplest machine learning models to those from this historical model it is hard to argue in favor of this hard coded algorithm. While we do not know how the performance of these models compare on a larger dataset we can already see that a trained machine learning model will outperform a hard coded algorithm that has to linearly traverse the whole dataset.

5.1.5 Mean squared error

The value of the networks Mean Square Errors is also of concern. All of the networks had high MSE, the reason for this is likely that the networks have a problem with predicting values consisting of high deviation from the norm. In simpler terms, the models are bad at predicting the edge cases. However, this is not the case for all of our investigated models. Inspecting the results from the two more advanced LSTMs reveals that their MSE, while still significantly higher, are closer to their MAE compared to the other models. This is likely due to the contextual memory of the LSTMs. By looking at the context of the data points close to each other in time, they can better predict larger deviations in arrival time; a definite advantage of LSTMs worth noting.

5.2 Challenges

5.2.1 Dataset

We acquired two datasets obtained from Ruter through Entur, public transit and vehicle monitoring data. We anticipated public transit data was the best source for approaching the problem. However, as the data proved to contain complete correspondence with estimated arrival times, it was no use to us and we had to use another source. The other option was the vehicle monitoring data set. This data was harder to work with as it was not structured right for this application. There was seemingly no rules to when a bus would rapport on its whereabouts, something that made defining labels and features a task potentially implying huge loss of potential data. After pulling the data, extracting the useful information, restructuring to fit the application, extract potential label-feature pairs, calculating first arrival, converting timestamps to sine-cosine pairs, adding weather data, and filtering out all but one bus stop pair, we were left with a questionable small data set to train on. In addition, we later decided that even though we had data from November 1. to April 30., we had to discard anything after February 24. to not confuse the models with abnormal data due to the COVID-19 pandemic that severely reduced the amount of data generated from buses. To be confident in the results from most of the models used in machine learning, sufficient data is needed. If the public transit data from Ruter had used real, recorded, arrival times, this would not have been an issue.

While the main focus in this paper is on contrasting various machine learning models, we used close to one fourth of our time gathering and processing the data. This includes searching in the data for useful information, developing methods for extracting and handling, and shaping the data without altering it. The computation of the data alone took days with efficient algorithms.

If this was to be attempted again in a different city with accurate public transit data, albeit fewer overall buses, it could result in a bigger dataset. Utilizing buses with Automated Passenger Counters (APC) would aid in depicting the world around the buses like our weather data currently does.

5.2.2 Models

Looking past the problems with the data set, there were other model-specific challenges we encountered that we had to solve. One of the first challenges we faced, was the fact that the models work in base 10 and do not comprehend the cyclic nature of things such as time. Even though this was an initial challenge, we managed to solve this by encoding time to be represented in the cyclic function of sine and cosine. This also brings us to the challenge we faced with the categorical data of weekdays. This obstacle was overcome by encoding the data as a one hot array of ones and zeroes.

Another challenge was encountered in the LSTM model. As the LSTM needs time series of data points as input, we needed to properly cut the data in a length that gives the model meaningful information without introducing noise from the unrelated data points. The act of finding this optimal time series length relies on both having accurate data for the context the time series is in, as well as being able to test and inspect what actual length is optimal. Because LSTMs takes a long time to train, and thus get meaningful data, it can be time-consuming to optimize this parameter, as well as all the others; a fact that also materialized in our work. The urban environment is stochastic with many factors that can be hard to accurately represent in a way the models can beneficially utilize. Events such as roadwork, accidents and deviations simply caused by an array of hard-to-predict human decisions are difficult to model and accurately portray. These combined factors prohibit a completely accurate model to be constructed. Indeed, as there would inevitably be some amount of uncertainty when dealing with such an active and unpredictable scene as a central urban environment. This uncertainty is likely part of the reason for our high MSE. An uncertainty element that we were not able to map in our features could lead our models to have a large margin of error in certain instances. This complexity and inherent unpredictability can be somewhat compensated for by using contextual data and memory as the LSTMs do, but there is still a noticeable error that can be attributed to this unpredictability.

Another unpredictable event that occurred during the period this research was carried out, was the outbreak of the COVID-19 virus. This virus caused incomprehensive lockdown of Norway, which substantially reduced overall road traffic and public transport. One of our goals was to compare the accuracy of our models against real-life arrival times. However, the pandemic likely had a widespread effect on factors impacting arrival times of public transport, perhaps suddenly transforming a major city to become more alike rural areas, transport

wise. Hence, the many uncertain factors and their potential validity-impacts led us to refrain from testing our models in this way.

In general, using machine learning models for predicting arrival times in an urban environment lead to better accuracy compared to a historical model. However, the computation times are massively increased, which is not ideal for a real-world application. Nonetheless, this increased cost must in turn be balanced against the fact that investments in the public transportation systems to make them efficient could also reduce emissions and thus greenhouse gasses and local pollution as well as lowering personal cost for the consumers.

5.3 Validity and reliability

As it stands, we have no major reason to doubt the overall validity and reliability of our findings, but several points should nonetheless be considered. First and foremost the quality of our data may cause concern, as we did not get accurate data containing exact arrival times of the buses, but rather a location with a timestamp that we had to use to approximate arrival times. Due to this, there could be some error in the data that could bring the validity of the accuracy of the models into question. We believe that a small margin of error in the data will not have a huge detrimental effect on the findings, as we did use location data that is also accurate. The second possible source of error is our extraction of data, outlined in the method section. This method relies on the fact that on average, there is no huge difference in time the buses takes to travel any part of the edge compared to another portion with the same length. It is unlikely that the time it takes to travel an edge is consistent across the entire edge all the time, but we believe that when looking at the average time across all our datapoints this will not introduce too high a margin of error. It should also be noted that all models were fed the same data, and such no model was trained on more accurate data than any other model. Due to this fact, the performance of the models relative to each other should be valid, even if their individual performance is affected by the data. When it comes to the reliability of our findings we have no notable concerns. If given the same data we used, using the same augmentation methods and the same models, we believe our data to be reliable and reproducible under the same circumstances. The cross validation of the results show no major discrepancies that give any reason to question the reliability of their results, and thus we believe them to be reproducible.

6 Conclusion and future work

There are many different models which may fit the purpose of predicting arrival times for buses in an urban area. We opted to research the use of SNN, ANN, SVR and LSTM. ANN has consistently been found a stable and rapid, although not always an accurate model for this problem, whereas prior research has indicated that SVR and LSTM both have the potential of improving accuracy, but with a training time cost. Using vehicle monitoring data from Entur, we conclude that optimizing the models, even though it is time consuming, is vital for predicting with better accuracy, especially with ANN and SVR. In particular for the LSTM model we detected a positive trend for the calculation times when tweaking and arriving at the optimal number of nodes and layers. For any real-world application it would be paramount to optimize the models for them to perform accurately and rapidly.

There were several challenges we faced and had to overcome when using vehicle monitoring data from an urban environment for our machine learning methods. The data itself proved challenging to work with, as such we had to use an alternative extraction method to produce enough feature label pairs. The data was also not in a optimal form for training machine learning models, as it had to be properly encoded. Urban environments are prone to factors that are difficult to properly represent as a feature, as such there are factors affecting traffic flow and travel time on an edge not properly learned by the models, resulting in inaccurate predictions, especially if these factors greatly affect the travel time.

The results accumulated shows that all four machine learning methods (SNN, ANN, SVR and LSTM) outperformed a historical data model. LSTM gave the lowest MAE at 57 seconds and therefore fared the best, prediction wise, followed by the advanced ANN, SVR, and the SNN giving the worst results with an MAE of 90s. These results do not concur with our hypothesis described in section 3.1 where we believed ANN would yield the best accuracy. Even so, these figures imply that although LSTM proved more accurate than the others, the improvement should not be overrated as it had a higher cost to train. These results concurs partly with our hypothesis and also with the overall conclusion that can be drawn from the previous studies (see Section 2); substantial differences between the machine learning methods cannot be found and historical models only outperforms machine learning models in rural areas. We can conclude from our work that all machine learning methods tested accumulated far better results than the historical data model in an urban area. Finally, given the large amounts of datapoints required for accurate predictions utilizing machine learning methods, the training time is substantially greater than using historical models. This fact should be taken into account when implementing the models to a real-world application. Even so, the improved accuracy provided by the machine learning methods may in many cases outweigh the costs of time use, making them the method of choice in urban areas.

6.1 Suggested future work

The list below are suggested future work:

1. Implementation of Kalman filter and compare its performance to the models in this study.
2. Investigate different kernels used in the SVR model in order to fit the purpose of the model better. This being better calculation speed and more accurate predictions.
3. The historical data provided from Entur was not optimal. Investigate if better data can be provided and see if that will gain better results.
4. The LSTM model provided the best results. However, there are more possibilities with using this model together with different kinds of layers. Investigate if the use of convolutional neural networks will improve the results as concluded in (Petersen, Rodrigues, and Pereira 2019).
5. Further increase the data by implementing traffic-flow and over all traffic speed gathered by either google or the Norwegian Public Road Administration (NRPDA). Investigate if these types of data will further improve the predictions of the models.

A Acronyms

ANN Advanced Neural Network

APC Automated Passenger Counter

AVL Automatic Vehicle Location

ITS Intelligent Transport Systems

LSTM Long-Short Term Memory

MAE Mean Absolute Error

MET The Norwegian Meteorological institute

MSE Mean Squared Error

SNN Simple Neural Network

NTNU Norwegian University of Science and Technology

ReLU Rectified Linear Unit

SGD Stochastic Gradient Descent

SVM Support Vector Machine

SVR Support Vector Regression

SYNOP Surface Synoptic Observations

Referances

By appearance

Festag, A. (2014). "Cooperative intelligent transport systems standards in europe". In: *IEEE Communications Magazine* 52.12, pp. 166–172.

Schrank, David, Bill Eisele, and Tim Lomax (2019). *2019 URBAN MOBILITY REPORT*. URL: https://static.tti.tamu.edu/tti.tamu.edu/documents/mobility-report-2019.pdf. (accessed: 20.04.2020).

Beaudoin, Justin, Hossein Farzin, and Cynthia L. Lawell (Jan. 2013). "Urban bus arrival time prediction: A review of computational models". In: *International Journal of Recent Technology and Engineering (IJRTE)* 2, pp. 164–169.

Carrel, Andre, Anne Halvorsen, and Joan L. Walker (2013). "Passengers' Perception of and Behavioral Adaptation to Unreliability in Public Transportation". In: *Transportation Research Record* 2351.1, pp. 153–162. DOI: 10.3141/2351-17. URL: https://doi.org/10.3141/2351-17.

Jeong, R. and R. Rilett (2004). "Bus arrival time prediction using artificial neural network model". In: *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749)*, pp. 988–993.

Fabrikant, Alex (2019). *Predicting Bus Delays with Machine Learning*. URL: https://ai.googleblog.com/2019/06/predicting-bus-delays-with-machine.html. (accessed: 27.01.2020).

Entur, https://www.entur.org/om-entur/ (n.d.). URL: https://www.entur.org/om-entur/. (accessed: 27.01.2020).

Williams, Billy M. and Lester A. Hoel (2003). "Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results". In: *Journal of Transportation Engineering* 129.6, pp. 664–672. DOI: 10.1061/(ASCE)0733-947X(2003)129:6(664).

Altinkaya, Mehmet and Metin Zontul (Jan. 2013). "Urban bus arrival time prediction: A review of computational models". In: *International Journal of Recent Technology and Engineering (IJRTE)* 2, pp. 164–169.

Petersen, Niklas C., Filipe Rodrigues, and Francisco C. Pereira (Apr. 2019). "Multi-output bus travel time prediction with convolutional LSTM neural network". In: *Expert Systems with Applications* 120, pp. 426–435. DOI: 10.1016/j.eswa.2018.11.028.

Zou, N., J. Wang, and G. Chang (2008). "A Reliable Hybrid Prediction Model for Real-time Travel Time Prediction with Widely Spaced Detectors". In: *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pp. 91–96.

Amita, Johar, Jain S. Singh, and Garg P. Kumar (Dec. 2015). "Prediction of bus travel time using artificial neural network". In: *International journal for traffic and transport engineering* 5, pp. 410–424. DOI: 10.7708/ijtte.2015.5(4).06.

Patnaik, Jayakrishna, Steven I-Jy. Chien, and Athanassios Bladikas (Mar. 2004). "Estimation of Bus Arrival Times Using APC Data". In: *Journal of Public Transportation* 7. DOI: 10.5038/2375-0901.7.1.1.

Bin, Yu, Yang Zhongzhen, and Yao Baozhen (2006). "Bus Arrival Time Prediction Using Support Vector Machines". In: *Journal of Intelligent Transportation Systems* 10.4, pp. 151–158. DOI: 10.1080/15472450600981009. URL: https://doi.org/10.1080/15472450600981009.

Wu, Chun-Hsin et al. (Oct. 2003). "Travel time prediction with support vector regression". In: *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems* 2, pp. 1438–1442. DOI: 10.1109/ITSC.2003.1252721.

Olson, David L. and Dursun Delen (2008). "Support Vector Machines". In: *Advanced Data Mining Techniques*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 111–123. ISBN: 978-3-540-76917-0. DOI: 10.1007/978-3-540-76917-0_7. URL: https://doi.org/10.1007/978-3-540-76917-0_7.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

Simard, P. Y., D. Steinkraus, and J. C. Platt (2003). "Best practices for convolutional neural networks applied to visual document analysis". In: *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* Pp. 958–963. DOI: 10.1109/ICDAR.2003.1227801.

Duan, Y., L.V. Yisheng, and F. Wang (2016). "Travel time prediction with LSTM neural network". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1053–1058. DOI: 10.1109/ITSC.2016.7795686.

Agafonov, A. A. and A. S. Yumaganov (May 2019). "Performance comparison of machine learning methods in the bus arrival time prediction problem". In: *Information Technology and Nanotechnology: Data Science* 2416, pp. 57–62.

Chien, Steven I-Jy, Yuqing Ding, and Chienhung Wei (Aug. 2002). "Dynamic bus arrival time prediction with artificial neural networks". In: *Journal of Transportation Engineering* 128, pp. 429–438. DOI: 10.1061/(ASCE)0733-947X(2002)128:5(429).

SYNOP Codes, https://orap.met.no/Kodeforklaring/Kodebok/koder/ww.html (n.d.). URL: https://orap.met.no/Kodeforklaring/Kodebok/koder/ww.html. (Timestamp: 15:21-16:26. accessed: 09.05.2020).

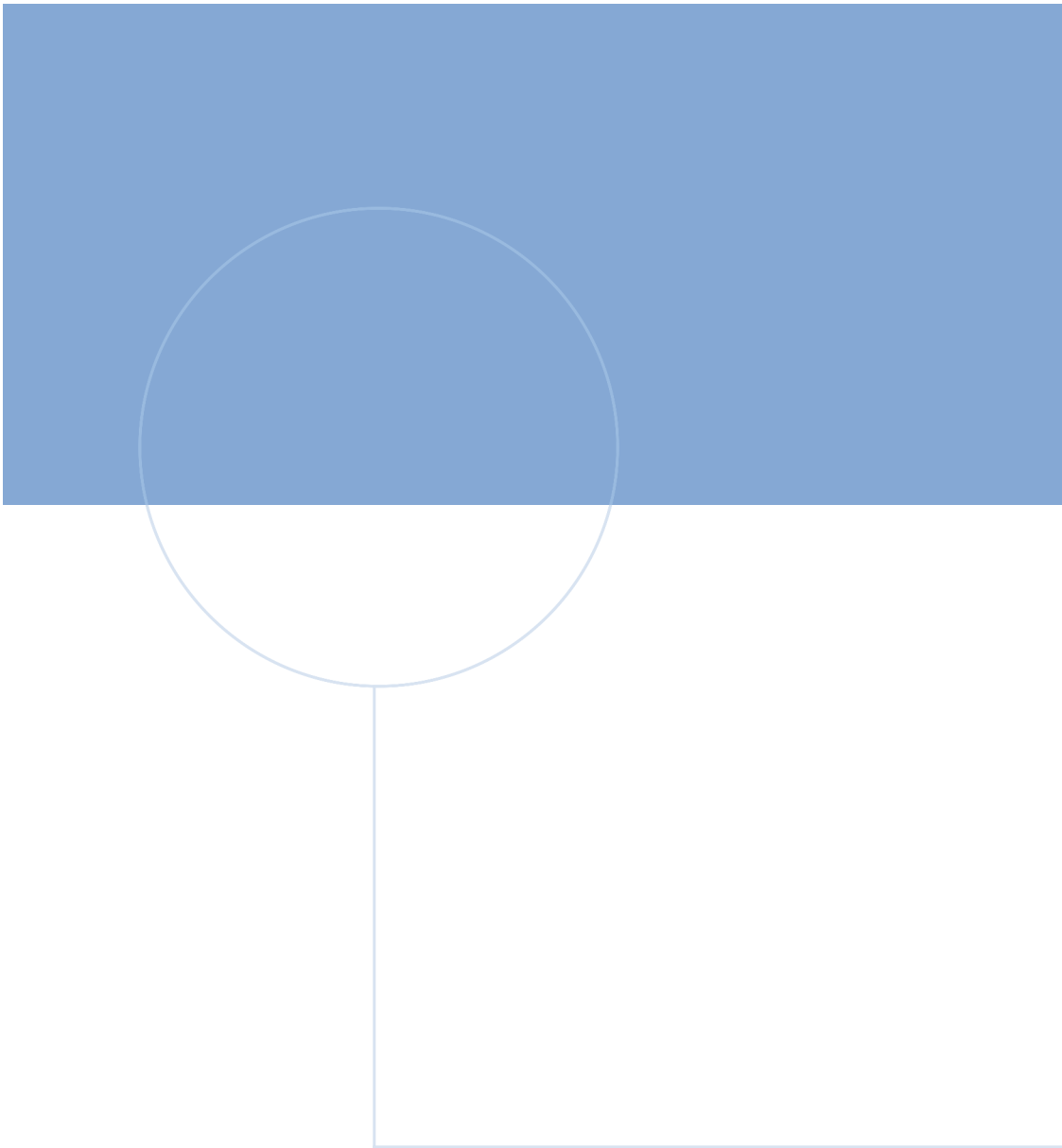
Kingma, Diederik P. and Jimmy Lei Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference for Learning Representations*. URL: http://arxiv.org/abs/1412.6980.

Genevay, Aude et al. (2016). "Stochastic Optimization for Large-scale Optimal Transport". In: ed. by D. D. Lee et al., pp. 3440–3448. URL: http://papers.nips.cc/paper/6566-stochastic-optimization-for-large-scale-optimal-transport.pdf.

Li, L. and G. Sanderson (2017). *But what is a Neural Network? — Deep learning, chapter 1, https://youtu.be/aircAruvnKk?t=1022*. URL: https:

//youtu.be/aircAruvnKk?t=1022. (Timestamp: 17:03-18:26. accessed: 04.05.2020).

Maker, Daemon (2014). *Answer to: What are the advantages of ReLU over sigmoid function in deep neural networks?*, *https://stats.stackexchange.com/questions/126238/what-are-the-advantages-of-relu-over-sigmoid-function-in-deep-neural-networks*. URL: https://stats.stackexchange.com/questions/126238/what-are-the-advantages-of-relu-over-sigmoid-function-in-deep-neural-networks. (accessed: 10.05.2020).



NTNU

Kunnskap for en bedre verden