Utilizing Mixture Density Networks for Travel Time Probability Distribution Predictions

Holm, Mikkel Elkjær
meh15@student.aau.dk

Kirkeby, Laurids Vinther
lkirke15@student.aau.dk

Department of Computer Science, Aalborg University, Denmark

June 12, 2020

Abstract

This paper concerns prediction of travel time distributions for road segments. The approach is a mixture density network, which is a combination of deep neural networks and mixtures. This approach allows better representation of travel time distributions than more traditional methods that only consider the mean travel time. The data set used in this paper is provided by a research group from Aalborg University consisting of a large amount of floating car data. The data is pruned, map matched and reduced to a topological road network. The model is trained with special features such as points of interest and neighborhood embedding. This has helped the model achieve a low negative log likelihood score. This model can then be used to generate routes between origin-destination node pairs using various pathfinding algorithms.

1 Introduction

Travel time prediction is an important tool in several different applications, such as route finding, congestion analysis and traffic forecasts. Traditionally, travel time prediction was done with simple mathematics, such as multiplying the speed limit with how far was needed to be traveled on road, and added time for what was the expected waiting time in intersections. Today, as travel time data has become, plentiful it is possible to create statistics and travel times for each road. However, the old method is often defaulted to when traveling on a road where little or no data exists for.

With the advent of powerful machine learning tools such as deep neural networks it is possible to create models for predictions of travel time. This increased the precision of the travel time prediction greatly[1] and made it possible to predict travel time for roads where little data exists. When creating routes that use the travel time predictions, it can be useful to use a probability density function of the travel time rather than a single mean value. The idea is that roads where the variance is low are preferable to roads where the variance is high. The lower variance guarantees more accurate travel time, which is useful in real life scenarios where there may be another route that is faster on average but has a higher variance. Given a deadline,

the lower variance route can be preferable in order to eliminate risk of being late.

In this paper we propose a system that is able to train a model for travel time prediction from a float car data set, utilizing novel features such as distance to point of interests. The system uses the model to make travel time predictions on a given road network, and use those predictions to make routes from point a to point b.

The rest of the paper is structured as follows. Firstly, there is some related work to the problem, as to help create a basic understanding of the problem. This is followed by the problem statement and the proposed framework for the system, as well as the theoretical background for the predicting and route finding. Then some experiments where our solution is tested against common machine learning models, features are evaluated and the routes are tested. Finally, we will discuss the results of the experiments and conclude on the paper, as well as describe what could be improved on in the system in the future combined with other interesting future works.

As a motivating example of why a probability distribution can improve travel time predictions, especially when there is a deadline for arrival, see Figure 1.

The probability distribution of road 1 has a faster mean travel time than that of road 2, but the variance is much higher. However, road 1 has a risk of being slower than road 2. Hence, road 2 provides a more reliable travel time, as its mean is much more frequently achieved, which can be crucial when creating routes that require a lower uncertainty threshold. Furthermore, in certain situations like for lorry drivers, the reward of arriving a bit earlier due to taking a route with high variance is not necessarily worth the risk of being late, as they can only drive for a certain amount of hours.

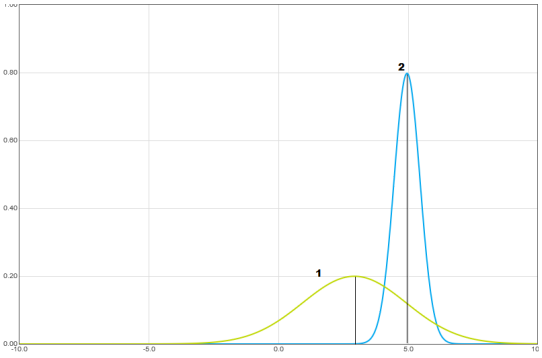

Figure 1: Travel time probability between two roads of equal length

A traditional machine learning model is not able to capture the variance and standard deviation and would only get the mean whereas by predicting probability distribution function with a mixture density network we are able to overcome that limitation while still being able to predict precise mean values.

1.1 Related Work

The problem of accurately predicting estimated time of arrival (ETA) is widely studied in the Geographic Information System (GIS) community and two major categories of solutions have been established, the model-based solutions and the data-driven solutions[2]. It should be noted that even though predicting and estimating travel time is often used interchangeably there is an important difference. Estimating travel time is trying to find the travel time of an already travelled route. Predicting is trying to find the travel time for routes for which the data is not available yet. We are trying to do the latter.

An example of a simple model-based so-

lution for predicting travel time is to use an intuitive physics based solution by dividing and summing the distance and speed limit of each road segment and add the expected waiting time at each intersection of the path. This follows the basic idea behind all route-based solutions i.e. divide the travel time finding problem for the whole route into sub-problems of finding the travel time for each segment and the time waiting at intersections. This could also include finding the average speed for a route segment. The simple approach operates under a number of naive assumptions; (1) the driver will keep pace with the speed limit of each road segment; (2) the expected waiting time at each intersection is static; (3) external factors such as traffic and weather are non-influential. Other approaches include to model the kinematics of traffic flow and queuing behavior to predict how the travel time will be under different traffic assumptions[3][4].

To overcome some of those problems, machine learning is often used in model-based solution by creating a model for predicting the travel time for each route segment or in a intersection by using historic data and regress on it. There are several different methods for doing this, but all come with certain drawbacks. Firstly, spatial temporal data tends to be sparse over the whole road network, and will often not be enough to get the real time traffic patterns for the whole network. Secondly, as traffic patterns are dynamic it is not a guarantee that a good prediction for the same day under the same conditions are the same. Thus it can be hard to predict the correct travel time even for a model. Furthermore, as the problem have been divided into smaller sub-problems there are a greater amount of accumulation of errors in the final prediction. And lastly, different people drive differently and without personalised data it can be hard to predict how a user behaves in reality, which can have an effect on the travel time and the traffic flow as a whole.

With the advent of mass tracking of cars and traffic it has become possible to look at historical data to statistically predict the future and detect travel time patterns. This is the so called data-driven approach. At the most general level, a data-driven approach could be to take all the historical data for a given road and average the travel time. There are a number of problems with this as it does not take all factors into account, such as difference of day, weather, season etc. It assumes that for any given day, the travel time is static. Machine learning models help with this, as it is possible to use many different features and real time data to make a prediction on a road under different conditions. Common models for predicting travel time include models that are parametric such as linear regression[5], Kalman filters[6], and models that are non-parametric, such as different deep neural network[7][8]. Recently, there have been efforts in predicting the travel time probability distribution to capture a whole range of possible travel times and how likely they are, rather than a simple mean value[9]. In Travel-Time Prediction Methods: A Review[1], they found that the non-parametric solutions worked the best as they had the lowest mean average percentage error In our own work Estimating Travel Time Probability Distributions using a Mixture Density Network[10] we explored different state-of-the-art ways of predicting travel time probability distributions and found that of the neural networks, Bishop's Mixture Density Network[11] was the best fit for us. When creating a model

for travel time there are two main goals either predicting for a whole route or predict it for each road segment and piece them together as the route grows. The problem with the former is that, it has a hard time predicting travel time for routes that are not recorded in historical data, routes of radically different lengths than the historical data could have a large impact on the preciseness of the prediction, but it does not get effect by the accumulation of errors that the latter method would. The latter on the other hand have the advantage of being able to create a general model for a road, but requires a large amount of diverse historical data and a decent cover of the road network for which the model is created. We have chosen to go with trying to predict the travel time for each road segment, as we believe that the accumulation of errors can be overcome and that a general model for roads is more advantageous.

1.2 Problem Definition

Based on the introduction and the related work we define the following problem definition:

Given road graph $G(N, E)$, the map matched data set $\mathcal{D} = \{R\}$ and augmented data Arg, is it possible to learn and predict the travel time distribution $P(td|e, Arg)$, where td is the travel time distribution, and $e \in E$ in such a way that it is possible to create a route $r = [e_i]$ utilizing that data?

Where the terms are defined as follows:

Street network: A street network is represented by a directed graph $G(N, E)$, where N is the nodes of the graph representing intersections and E is the edges between the nodes representing road segments.

Observations: An observation $o = (p, t, v)$ is a GPS recording, where p is the longitude and latitude, t is the timestamp and v is the velocity of the car.

Trip: A trip $T = [o_i]$ consists of a number of consecutive observations.

Map matched data: Map matched data is calculated based on a map matching algorithm and each point is matched directly to a road segment. It is given by $R = (T, o, d, tt, rt, d)$, where T is the trip and contains an ID, which the data belongs to, o and d are the origin and destination nodes, respectively, where $o, d \in N$. tt is the travel time, rt is the recorded time of day and d is the distance traveled.

Augmented data: Augmented data denoted by Arg are the features that describe the road network data, this includes point of interests (POI), road type, max speed, network embedding of the road edge and its neighbours and neighboring data.

2 Approach

2.1 Data Processing

Name	Description	Datatype
TripId	Trip identifier	BigInt
BoksId	Car identifier	BigInt
LocalDate	Local date	Date
LocalTime	Local time	Time
Lon	Longitude	Real
Lat	Latitude	Real

Table 1: The floating car data (FCD) table schema.

A data set of 1.1 billion floating car data

for cars driving in Denmark over the period April 2012 - December 2014 was obtained from an ongoing research project at Aalborg university. The format of the data can be seen in Table 1. To be able to use the data for the purpose of this project, some modification of the data was needed. Firstly, each data point was map matched to a road network retrieved from Open Street Maps (OSM). The map-matching itself was done by using the Open Source Routing Machine (OSRM), which is able to return map-matched data points as an array of route objects, which in sequence creates a route. Each route object has a confidence interval of map-matching between 0 and 1. Route objects are only kept if their confidence interval is above 0.85, as it otherwise can be too unreliable. Routes that had a route object disregarded was short-circuited to the next route element that had an good confidence interval. Each route object is further divided into route legs which have the form $(w_1, w_2, w3)$, which represent a route between two way points. These have the node property from OSM. Thus, the last step was to divide all the route legs into their pair data e.g. (w_1, w_2), which represents an edge from waypoint 1 to 2 and record the time it took from when the car passed waypoint 1 to waypoint 2. The data size is reduced further by creating a topological road network graph that merges edges, that do not end or start in a intersection. Therefore, all elements in a road network from intersection i to j could be merged and their travel time added together. At the end of the data processing, there are only 42.5 million of data-points in the form of Table 2. This is a significant reduction of the starting data set, which also makes it significantly faster to train on. A full flowchart of the process can be seen in Figure .

Table 2: The edges table schema.

Name	Description	Datatype
TripId	Trip identifier	BigInt
BoksId	Car identifier	BigInt
Source	Starting node	BigInt
Destination	Ending node	BigInt
TravelTime	Travel time	Double
DateTime	Date and time	Timestamp
Distance	Distance	Double

2.2 Feature extraction

To make precise travel time distribution predictions, it is important to have a rich set of appropriate features while not so large that some of features detract from the ability of the model to learn and predict[12]. It is also important to select features so that the data leakage problem does not occur[13]. We have selected three major categorize of extracted features: spatial, temporal and augmented.

Spatial features: A major influence on the prediction of the travel time distribution is geospatial and structural attributes of the road segments. One of the most important features here is the length of the road, as it is linearly linked to the travel time of the road, this can be seen in correlation schema in Figure , where there is almost a 1 to 1 correlation. Another feature is how far away a specific point of interest is from the road edge. These points of interest could be facilities with a high influx of traffic, such as schools or office buildings, this can be used to enhance prediction, as congestion is a factor for travel time at different times of the day. Furthermore, virtually identical roads can vary widely in travel

time based on where they are located, such as close to important POI or out in the country side. We propose that this can be captured by using the distance to the nearest POI as a feature. The length of an road edge is retrieved from the data set. The point of interest are obtained by choosing areas of high influx of traffic, figuring out how far an edge is from the nearest point of interest of that type and saving it. To do this we have made use of the API Overpass Turbo[14], which is able to retrieve nodes that have a specific type of POI. The POI types can be found in OSM[15]. The API calculates the distance from the POI to the nearest road edge. This is a fairly costly process but only needs to be done once for each POI. In this project we have used hospitals and schools as POIs, as they have a large amount of in and out traffic.

Temporal features: Another influence on the travel time distribution is the temporal information of when the road segment was driven on. There are two features of temporal information that can be extracted, namely rush hour or not rush hour. Additionally, there is the season for when the record of FCD was taken. Rush hour or not has an effect because of the increased risk of congestion, the choice was made to keep it at only two values instead of every hour of the day, as the two definitions of time of day represent thoroughly represents the traffic patterns. The season has an effect, as weather and road conditions vary widely based on the season of the year. In the winter months for an example, the road can be icy and thus force the driver to drive slower, similar in the autumn rain can slow down the driver, while in summer the conditions are more optimal. The information is gathered from the timestamps of travel trajectories where the hours 7-9 and 15-17 are marked as rush hour[16].

Augmented features: Augmented features is all additional features that can be retrieved based on the data from outside sources, which are not included in the above two classes, rather than from the road trajectories themselves. As the speed limit dictates how fast it is legal to drive on a road, it will have great influence on the travel time. As the speed limit is important, so is the type of road, as some types of roads are more prone to congestion than others and some are designed to let the traffic flow more smoothly. Another feature is the roads' neighboring roads, as the speed limit and type etc. of adjacent roads have a great effect on the travel time for the road. Recently, there have been achieved results by using node2vec[17], which takes a number of random walks around a node of a predetermined length and embeds those routes using a skip-gram model much like word2vec embeds sentences. node2vec is used to transform a node into an embedding vector as a way to capture its surrounding network's structure. They found that even though node2vec is designed for capturing structural information of social networks, it could be used to some extend for embedding roads, which helped with classifying roads. As such we have chosen to not only embed the road with node2vec but also the neighboring roads for the same reason as to why the adjacent routes features are important for the prediction. The max speed and road types have been found by matching the recorded edges with the OSM network of Denmark. The embedding was done by using node2vec for each node in the graph for the road network of Denmark and then summing the two nodes that represent an edge's vectors together and save the resulting vector to the edge.

2.3 Mixture density network

The basic idea behind a mixture density network is to modify a deep neural network in such a way that it is able to capture multi-modal probability distributions. This meaning that the DNN provides the parameters for the different distributions, which is then mixed together by some weight that the DNN also provides[11]. The overview can be seen in Figure 2, which has input vector **X**, a single hidden layer which outputs the distribution parameters, and the mixture parameter which mixes the distributions and outputs a vector of the mixed distributions. The advantage of MDN is the ability for it to predict multi-modal distributions rather than just a single Gaussian distribution[11].

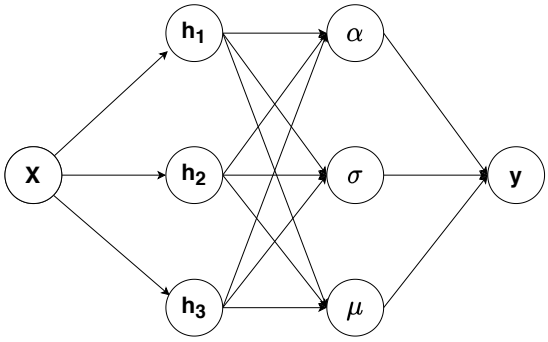

Figure 2: Mixture density network overview

The conditional probability of a mixture is defined as:

$$p(y|x) = \sum_{c=1}^{C} \alpha_c(x)\mathcal{D}(y|\lambda_{1,c}(x), \lambda_{2,c}(x), ...) \tag{1}$$

c denotes the mixture index. The mixture index depends on the assigned amount of components. More than one component results in a multi-modal output.

α denotes the mixing parameter, which is conditioned on the input x.

$\mathcal{D}$ denotes the distribution to be mixed, in this case there are two output elements to be mixed.

λ denotes the various parameters of $\mathcal{D}$ for which there can be multiple, as in this case the interest is in finding the PDF for a Gaussian distribution, as such there are two parameters μ and σ. If it was a truncated Gaussian there would be up to four parameters.

λ_1 denotes to the conditional standard deviation $\mu(x)$.

λ_2 denotes the conditional variance $\sigma(x)$.

To summarize, it calculates how likely the value y is to be drawn from a mixture distribution $[\mathcal{D}$.

A single layer deep neural network (DNN) with ReLU activation is modified.

$$h_1 = max(W_1^\top x + b_1, 0) \tag{2}$$

$$\alpha(x) = softmax(W_\alpha^\top h_1(x) + b_\alpha) \tag{3}$$

$$\lambda_1(x) = (W_{\lambda_1}^\top h_1(x) + b_{\lambda_1}) \tag{4}$$

$$\lambda_2(x) = nnelu(W_{\lambda_2}^\top h_1(x) + b_{\lambda_2}) \tag{5}$$

Where W is the weights and b is the bias. As can be seen the hidden layer is used as input in the output parameters. λ_2 has a non negative activation function as it must not become negative as a Gaussian requires a positive standard deviation.

$$\arg\min_{\Theta} l(\Theta) = -\frac{1}{|\mathbb{D}|} \sum_{(x,y)\in\mathbb{D}} \log p(y|x) \tag{6}$$

The loss function used to train the model and thus back propagate change is the negative log likelihood (NLL), which takes the

negative log of the conditional probability as seen in Equation 1.

2.4 Route Generation

Creating routes from the graph is a shortest path problem. Depending on the size of the graph and other requirements for the system, various algorithms can be applied for pathfinding. A* with an admissible heuristic or contraction hierarchies can be used to find great routes on larger graphs, whereas algorithms like Dijkstra's can be used if the absolute best route is required, at the cost of higher a computational complexity. The main scope of this project is not pathfinding algorithms but rather the predictions of travel time distributions. Thus, the testing of the model will be conducted using a simple implementation of Dijkstra's algorithm. This ensures that the best route prediction based on the model is selected for comparison.

2.5 Architecture

Based on the feature extraction, route creation and theory we propose the system illustrated with a flowchart in Appendix B.

The system is divided into two parts. A machine learning module that takes historical floating car data, extracts features, trains a model and creates a travel time augmented graph. The second part is a route finding module which uses the graph together with time information to create a route given by a start and an end point.

2.5.1 Machine learning architecture

The machine learning module have been expanded in Figure 3

From the top, a selection of POI categories are selected and passed for coordinate retrieval from the Overpass Turbo API. Afterwards, each edge of the road network, which is retrieved from the Open Street Map, is iterated through and the nearest POI in each POI category is found and saved to the road graph. Then each road is embedded by using word2vec, the embedding is saved to the graph. When that is complete, it is possible to get the neighboring embedding by finding the neighborhood of the edge and summing the embedding vectors. At the same time, the features of the edge's neighbours are also

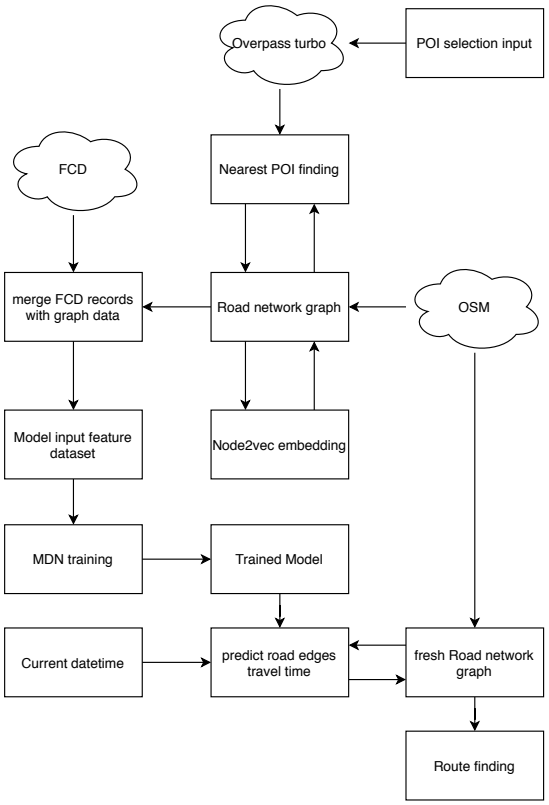

Figure 3: Machine learning architecture that takes a dataset of floating car data and a list of Point of interests and transforms the FCD into trainable features and augments them with outside information and embedding of the graph. The output is a road network graph with a predicted travel time probability distributions

found and added to the edge. From a database, the floating car data is retrieved, cleaned and filtered so that it can be used as features. Each row in the FCD data set is then matched up with their matching edge in the augmented road graph and the graph variables are used as additional features. The label is extracted from the data set and the model is trained and saved. The model is then used to predict the travel time probability distributions, which are saved to a fresh road graph and sent to the route finding module.

3 Experiment

The experiments are divided into two different parts experiments on the machine learning and experiments on the route generation.

3.1 Machine learning experiments

We evaluate the accuracy of the machine learning part and its features on a set of real world travel trajectory data of traveling cars equipped with GPS and tracked over several years. We evaluated our data set and how the features we selected are related in it. We also test our model with different mixtures against a null model, a linear regression, and a deep neural network that are modified in such a way that they also can predict a travel time distribution. Furthermore, we conduct an ablation study to find out how well our selection of features help the model predict the travel time distribution.

3.1.1 Data set

The data set is comprised of 1.1 billion different data rows of floating car data from all around Denmark, retrieved from an ongoing research project at Aalborg university. The data is processed into a format that can be used for training as described in section 2, such that there only are 42.5 million rows of data. The data is further filtered by removing rows where the travel speed is above 155km/h and under 1km/h as the values likely were erroneous and lessened the models prediction effectiveness. Roads that connect the different islands by ferry were also disregarded. After filtering, there is around 36.7 million rows to train on. To ensure that there is a good coverage of the road network of Denmark, the floating car data has been plotted into a heat map based on how often a node in the road network was visited in the data set. As can be seen in figure Figure 4, most of the floating car data is centered around northern Jutland more specifically Aalborg, but a wide area of the country is covered. More specifically, out of the 861 thousand edges in the road network for Denmark 212 thousand edges have been visited which is around a 25% coverage

To explore the data set to see how impactful the training features are, a selection of descriptive statistics for the features have been calculated and can be seen in Table 3. One hot features are disregarded, as their value can only be 1 or 0. In Figure 12, the correlation between features can be seen. Travel time, length and the neighborhood attributes have the largest standard deviation, which can be expected, as they have the largest range of values, as roads can be a wide variety of length and thus travel time which is also accounted for in the neighboring nodes value. More interesting is that there is almost a 1 to 1 correlation between travel time and length of a road, furthermore the maximum speed, neighborhood

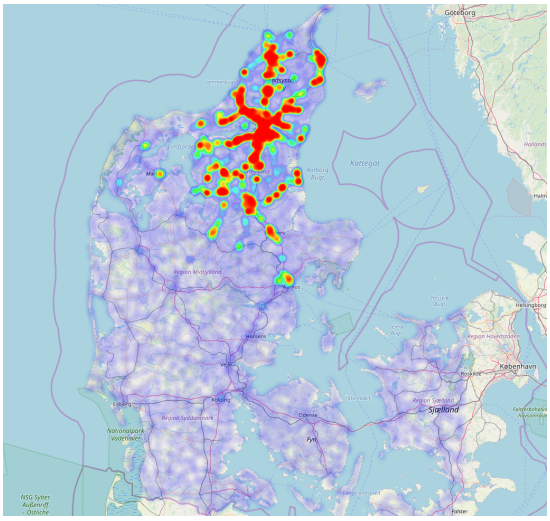

Figure 4: Heat map that denotes the number of times a node is visited by a driver. The more visits the more red a node is

attributes and motorways have a large correlation and schools have a small correlation to travel time but is still noticeable. This can be attributed to that schools are in areas with high traffic influx.

3.1.2 Experiment Setup

The linear regression, DNN and MDN model were implemented in Python using tensorflow 3.0 and trained on a single Nvidia 980ti GPU on a system with 16Gb of RAM. The travel time is used as the training label, and the holdout method was used where the data was divided between 80% training and 20% testing data. 20% of the training data is used for a validation set. The model was trained with batch of size 35000 over 500 epochs or until the difference between training gain of the epochs became very small. As the size of the data set is large at around 4.3Gb, the model was trained in chunks of the dataset at a time, each chunk was 8000000 rows long, as such the data was divided into 5 chunks, each with their own validation. In each layer of the model there are 200 nodes, the model has 2 layers of conventional deep neural networks before using 3 MDN layers in parallel. Before chunking, the data was randomized as to lower over-fitting. A typical training took around 10 minutes and each chunk usually ended before 500 epochs was reached. Usually, it took 100 epochs to reach convergence for the first chunk and round 20 epochs for the remaining.

3.1.3 Loss function

The most common loss functions used to evaluate the models are the mean squared error and the negative log likelihood. MSE was chosen as it is very commonplace in machine learning and gives an intuitive comparisons between the different models and their performance. The negative log likelihood was chosen as the second loss value, as it calculates the probability that the test value should be drawn from a probability density mixture given the parameters generated by the network, as such if the negative log likelihood is high it is unlikely that the mixture generated by the model is correct.

3.1.4 Results

Table 4 shows the results of training the different models where the # is the number of mixtures used when training the model. As can be seen, all models perform better than the NULL model, which suggests that there is some relation between the features and the label feature. Furthermore, it can also be seen that the DNN and MDN outperform the linear regression model, which suggests a non-linear relationship between the features and the label as expected. This will further be proven by the ablation study

	count	mean	std	min	25%	50%	75%	max
traveltime	36723233	19.55	28.53	1.00	4.33	10.00	24.00	400.000
length	36723233	411.41	833.90	2.08	52.70	125.93	407.11	11380.40
hospital	36723233	18.02	8.97	0.02	12.51	19.92	22.72	66.21
embedding	36723233	2.05	0.16	0.17	1.96	2.06	2.15	3.68
school	36723233	4.26	3.84	0.01	1.22	2.76	6.42	22.62
max speed	36723233	75.20	19.25	10.00	60.00	80.00	80.00	130.00
neighborhood emb	36723233	2.03	0.13	0.52	1.97	2.05	2.11	4.86
neighborhood att	36723233	357.28	392.57	7.04	99.49	174.43	518.63	7375.00

Table 3: The table describes the data for the different features in the dataset. Count is the number non-null values, mean is the mean value of a feature, std is the standard deviation, min is the minimum of the values, the percentages describe the quartiles and max is the maximum of the values the feature ever takes.

later.

There is a slight advantage in the MSE in the DNN model over the MDN model of any mixture. This was expected since the MSE takes the mean value of the probability distribution, and as there are more peaks in a mixture it will be different from the single curve distribution. This can also be seen in the figure, as the one model with MSE close to the DNN is the MDN model with 1 mixture. It can also be seen in the table that MDN is better with a NLL loss-function, which is expected as it finds the probability that the predicted values are in a probability distribution, given by the predicted probability distribution function. It was found that 3 mixtures was the best when it came to NLL with a value of -0.017. If the number of mixture increases more there is a slight increase in NLL loss. In the further experiments, a mixture of 3 is used.

#	Model	MSE	NLL
	NULL	1.222	1.530
	LR	0.576	1.229
	DNN	0.082	1.005
1	MDN	0.095	0.061
2	MDN	0.105	-0.004
3	MDN	0.110	**-0.017**
4	MDN	0.115	0.030
5	MDN	0.121	0.002
6	MDN	0.122	0.009
7	MDN	0.121	0.023

Table 4: Loss function test results for all models where lower is better. The null model is created by sampling random values in the evaluation methods within some reasonable interval based on the test data.

Model	NLL	Δ
MDN_base	-0.017	0
MDN_length	0.410	0.427
MDN_time	1.497	1.514
MDN_neighbours	1.219	1.226
MDN_road	1.217	1.224
MDN_POI	1.054	1.061
MDN-v	0.148	0.155
MDN-pv	0.149	0.156

Table 5: Ablation study results, MDN_base is the baseline model with all features, Δ column is the difference in loss function. MSE loss are not used as the interest was in seeing how the model performs when predicting probability distributions

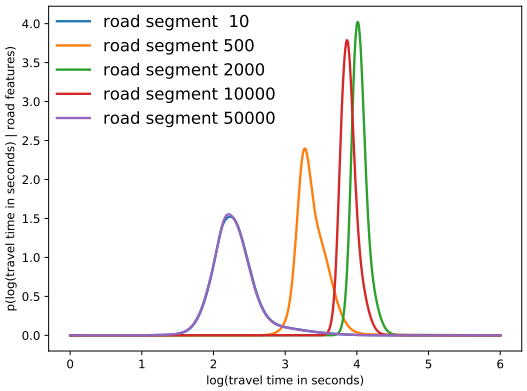

Figure 5: Probability distributions for five road segments. The x-axis denotes the travel time and the y-axis denotes the probability

To study the effect of the different features on the training, the models MDN_time, MDN_neighbours, MDN_road, MDN_length and MDN_POI were created, which are models with only that respective category of features. MDN-v is the model without embedding values and MDN-pv is the model without embedding values or POIs. Their loss values and delta change can be seen in Table 5. MDN_length has the smallest stand alone NLL loss, which would suggest as very close relationship to travel time as can also be seen in Figure 12. It can also be seen that POI is second, which would suggest that there is some relation that is not evident in the correlation table. Furthermore, we can see that no model outperforms the model with all features, which would suggest that no feature is redundant.

An example of the PDF predictions for 5 different roads can be seen in Figure 5 graphically where we can see that different roads have different variance and such a smaller probability for the mean value.

3.2 Route Examples

The following are comparisons of some routes generated from Origin-Destination pairs. These examples provide a simple visual comparison of the routes generated by the system when compared to a state-of-the-art tool.

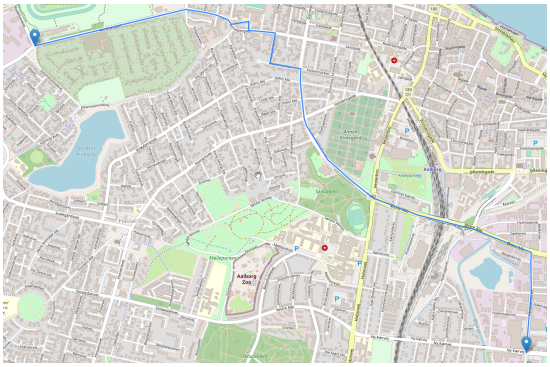

Figure 6: Example route 1 generated from the model

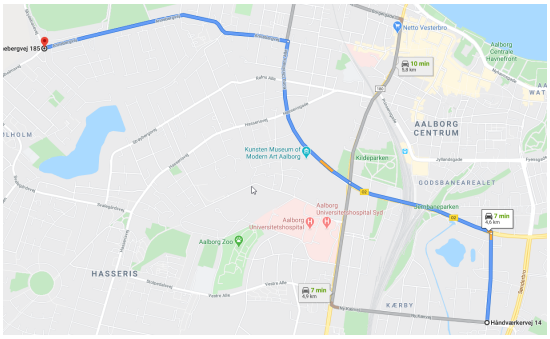

Figure 7: Example route 1 generated with Google Maps

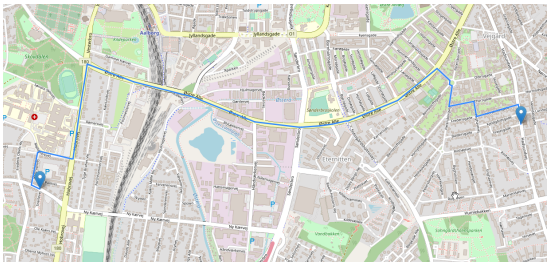

Figure 8: Example route 2 generated from the model

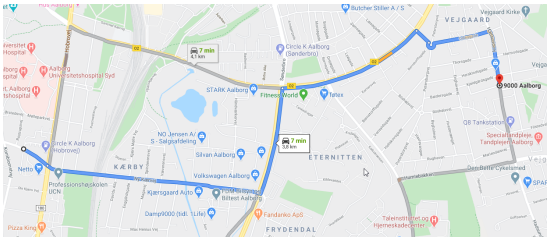

Figure 9: Example route 2 generated with Google Maps

For the first OD pair example, the routes generated were near identical to each other.

The second example is more interesting, as the routes vary a little bit from each other. The travel time estimation from Google is however the same, if the greyed out path is selected. The two systems produce fairly similar results for the given OD pairs.

4 Discussion

As can be seen in Table 4, a mixture with 3 components appears to yield the best results for the negative log likelihood, and are almost even with the DNN model in mean squared error. The reason for the small difference in result in the MSE for DNN and MSE can be attributed to that the MSE assumes a static standard deviation, which has been approximated for the MDN models. Therefore, it is not necessarily a good comparison fit, but can say a bit about how the mean values are predicted. The problem with different standard deviations can be seen as the MDN models; MSE worsens as the number of components and thus standard deviations increases. Another thing to note in the test is, that to be able to compare the different models to the MDN the variance and standard deviation was needed to be constructed outside their specific model. The way this was done was to use the predicted variance and std from the current MDN model, as such the biases and problems that the MDN would include in its prediction will necessarily also be present in the other models that utilize the values.

There is a risk that the aggressive data pruning done in subsection 2.1, which helps the prediction accuracy could remove too much of the data set to be truly representative of the whole road network. Most interesting would be the the pruning of data with a too small speed, which was added under the assumption of parking and pauses, but could just as well model queuing or red lights at a small road. By removing it, essential behavior and thus travel time could potentially be lost or simplified in the predictions. Especially small roads with lots of intersections would be lost by

the pruning. The upper speed on the other hand does not seem to be a big problem as some of the values that were pruned were unrealistic even when accounting for drivers not adhering to the speed limit.

The model is created under the assumption that years do not differ from one another e.g. May 1. 2019 does not differ in any way to May 1. 2020. In other words, the travel times are assumed to be static when considered from a year to year basis. This could be true as it would catch different holiday traffic but would not catch behavior under different weather, which also could have a huge effect on the travel time. The largest problem with utilizing so called real time data would be gathering them and would be time consuming and maybe not even very precise as weather in different parts of the country differ greatly.

Route Generation

It is worth noting that the nodes in OSM often exist in a specific lane which may have a direction, effectively forcing the route in one direction from the origin. This can lead to results of lower quality due to erroneous node selection when testing.

Since the Google Maps software is proprietary, it is hard to do a comparison of methodology. When it comes to travel time estimation, there is no competition, as Maps collects real-time location data from smartphone users to detect congestion and general speeds.

A challenge for this project is to properly test its predictions, as the system is predicting on an edge-basis. Most similar solutions out there predict travel times for whole routes, which makes it hard to compare. This system has complexities on an edge level they might not have and they will likewise have advantages on things that are not the main scope of this project. Furthermore, we have not been able to find any existing fitting benchmarks.

Finally, it should be noted that the model and the subsequent pathfinding may not necessarily be as proficient in predicting travel time in different countries as local traffic customs and weather in those countries. To fit the model to another country would need a new training data specific to that country.

5 Conclusion

We can conclude that we have successfully designed and implemented a travel time prediction system using a mixture density network with various interesting features. We were able to map match and reduce a large data set by 96.14% in order to speed up model training without losing important information. Furthermore, we have implemented an MDN with a wide array of good features which where able to predict the travel time probability distributions with a good NLL score of -0.017, which makes it very likely that the label values are accurately under the PDF. The model was also able to outperform baseline models with the exception of DNN, which was expected due to the characteristics of the MDN. We can also conclude that the features were useful, as removing them worsened the accuracy of the models predictions.

We were able to utilize the predicted distributions in pathfinding, which generated routes that appear similar to those generated by Google Maps.

6 Future Work

As discussed in section 4, the prediction assumes that the travel time for a given road at the same day for each year is static,

which is not always the case. To alleviate this problem, so called real time features could be extracted. These are features that differs from day to day and would rarely be the same at the same day in a different year. One of the features that could be included would be weather information, as it can have a great effect on the travel time. For example when it rains people are more susceptible to take the car to avoid the rain or when it snows the road are icy and thus travel time is longer than normal. To include this feature would require to retrieve the weather information for any day both historical and real time, in such a way that it is precise to the local weather at the road segment which would require access to the weather stations in proximity to the edge. This is at the time of writing available by accessing a large data set of the whole world by requesting access, but are limited to a number of requests per day. Alternatively, the data could be accessed from the Danish meteorological institute, which requires constant retrieval, as historical data is unavailable. Another thing that could be done in the future would be to experiment with different layers and structure of the model, as of now, there is only one hidden layer of 200 nodes between the input and output of the model. This could be experimented with different type of layers and nodes to see if it enhances the precision of the predictions. This could be done by taking inspiration of other state of the art methods that often merge different models with different structures and loss functions into a single model in order to improve the results. Lastly for the machine learning, it could be interesting to test our model against different state of the art methods. As it is very time consuming to construct these other models it has not been done in this project but could give a true measure of how the model performs.

In the pathfinding module future work would be experimenting with different kind of path finding algorithms other than Dijkstra to see how it would affect the route creation. An alternative would be to create a whole new algorithm, which incorporates the intrinsic values of probability distribution to route creation rather than using it in an ad-hoc method with the mean value and try to minimize the variance. The variance could also be better utilized to make confidence intervals of arrival time such that different routes could be create based on different confidence intervals e.g. some may be happy with a large risk of not arriving on time but have a potentially faster route.

Another future work would be to test the model by generating routes and comparing them to various routes generated by state-of-the-art tools. This could be done in different environments, under various conditions in order to see how the model performs. This can also be done with multiple shortest-path algorithm implementations for more interesting results.

As a last future work item could be to create an easy to use interface such that it would be possible for a user to navigate the system, where they could assign their preferred confidence intervals and input different start and end destinations. This would also be opportune to include different user profiles in the system because the optimal route is not always the fastest, as we have assumed in this project, but varies from user to user.

A Data processing flowchart

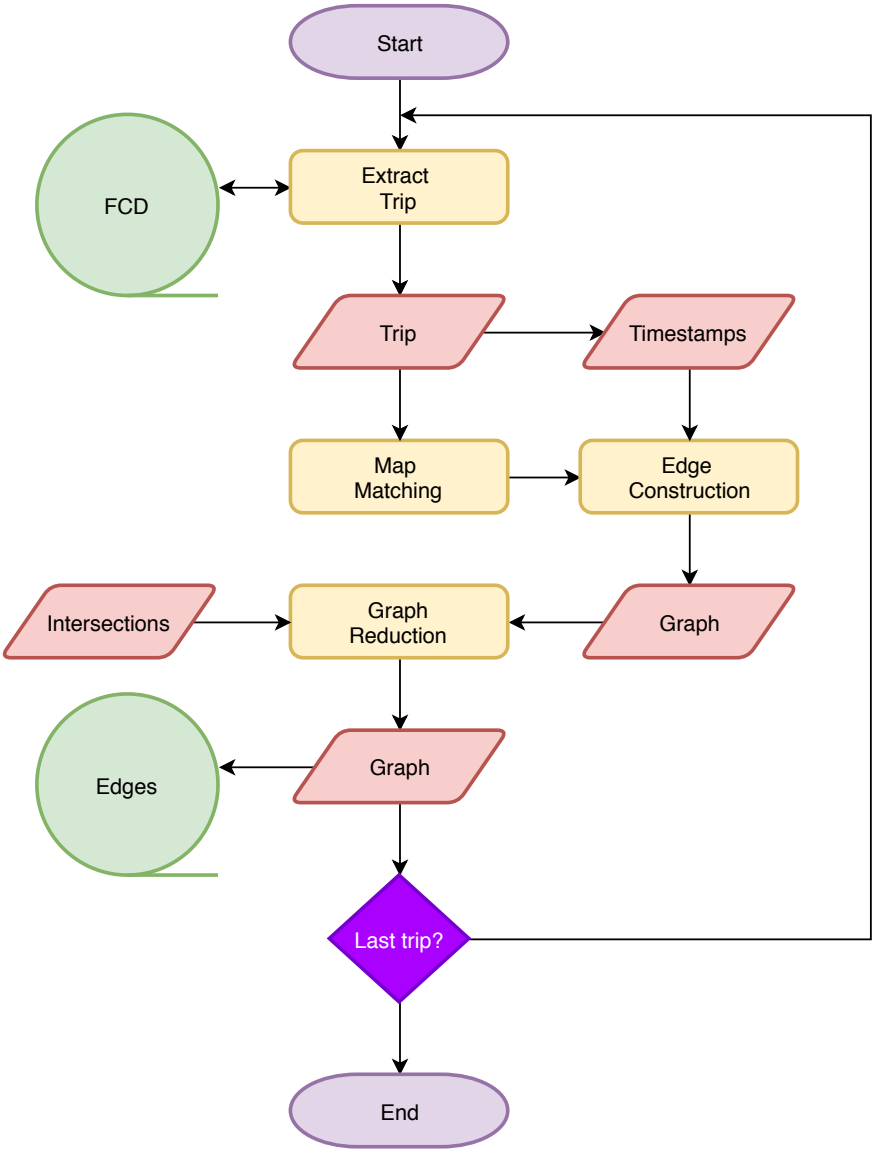

Figure 10: Data processing flowchart

B Information Flowchart

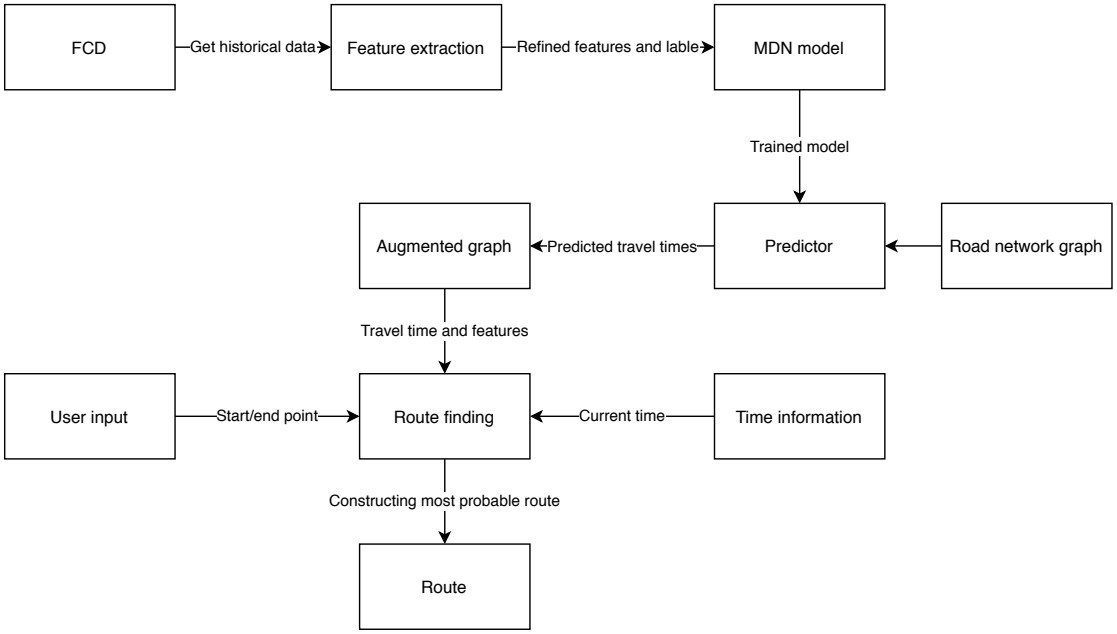

Figure 11: Overview of the flow of information in the system

C Correlation Table

	TravelTime	Month_autumn	Month_spring	Month_summer	Month_winter	DateTime_False	DateTime_True	Highway_motorway	Highway_primary	Highway_residential	Highway_secondary	Length	hospital	vector	neighbors	neighbors_att	Maxspeed	school
TravelTime	1.000000	0.002627	-0.009697	0.004440	0.002135	-0.022156	0.022156	0.435859	-0.016711	-0.085718	-0.226398	0.950686	-0.045087	-0.124201	-0.146610	0.565568	0.432753	0.086047
Month_autumn	0.002627	1.000000	-0.348546	-0.391603	-0.338714	-0.025270	0.025270	0.007620	0.003086	-0.004526	-0.004775	0.004214	-0.006847	-0.005200	-0.004715	0.004651	0.006091	-0.009745
Month_spring	-0.009697	-0.348546	1.000000	-0.322864	-0.279259	0.004520	-0.004520	-0.004844	-0.007952	0.005491	0.006060	-0.004508	0.002338	0.006709	0.005661	-0.005775	-0.004085	0.002042
Month_summer	0.004440	-0.391603	-0.322864	1.000000	-0.313757	0.040345	-0.040345	-0.003341	0.010427	-0.003503	-0.003658	0.003592	0.009647	-0.001285	-0.001242	0.008496	0.000539	0.013221
Month_winter	0.002135	-0.338714	-0.279259	-0.313757	1.000000	-0.019927	0.019927	0.000027	-0.006607	0.003247	0.003113	-0.003993	-0.005145	0.000367	0.000846	-0.008483	-0.003223	-0.005467
DateTime_False	-0.022156	-0.025270	0.004520	0.040345	-0.019927	1.000000	-1.000000	-0.046026	-0.005652	0.018606	0.024042	-0.024194	-0.026142	0.026347	0.023470	-0.025766	-0.041826	0.011953
DateTime_True	0.022156	0.025270	-0.004520	-0.040345	0.019927	-1.000000	1.000000	0.046026	0.005652	-0.018606	-0.024042	0.024194	0.026142	-0.026347	-0.023470	0.025766	0.041826	-0.011953
Highway_motorway	0.435859	0.007620	-0.004844	-0.003341	0.000027	-0.046026	0.046026	1.000000	-0.148471	-0.104061	-0.489413	0.529574	-0.081778	-0.316160	-0.336100	0.567791	0.705619	-0.032497
Highway_primary	-0.016711	0.003086	-0.007952	0.010427	-0.006607	-0.005652	0.005652	-0.148471	1.000000	-0.124412	-0.585123	-0.019112	-0.005361	-0.016239	-0.004493	0.028240	-0.075904	0.080569
Highway_residential	-0.085718	-0.004526	0.005491	-0.003503	0.003247	0.018606	-0.018606	-0.104061	-0.124412	1.000000	-0.410104	-0.113269	0.023790	0.146673	0.089645	-0.182727	-0.016328	-0.016689
Highway_secondary	-0.226398	-0.004775	0.006060	-0.003658	0.003113	0.024042	-0.024042	-0.489413	-0.585123	-0.410104	1.000000	-0.270766	0.044515	0.137290	0.174224	-0.292075	-0.399843	-0.029774
Length	0.950686	0.004214	-0.004508	0.003592	-0.003993	-0.024194	0.024194	0.529574	-0.019112	-0.113269	-0.270766	1.000000	-0.059516	-0.172837	-0.198805	0.604342	0.516978	0.082356
hospital	-0.045087	-0.006847	0.002338	0.009647	-0.005145	-0.026142	0.026142	-0.081778	-0.005361	0.023790	0.044515	-0.059516	1.000000	0.030513	0.031277	-0.060123	-0.070917	0.212926
vector	-0.124201	-0.005200	0.006709	-0.001285	0.000367	0.026347	-0.026347	-0.316160	-0.016239	0.146673	0.137290	-0.172837	0.030513	1.000000	0.284567	-0.196176	-0.253631	0.028364
neighbors	-0.146610	-0.004715	0.005661	-0.001242	0.000846	0.023470	-0.023470	-0.336100	-0.004493	0.089645	0.174224	-0.198805	0.031277	0.284567	1.000000	-0.242751	-0.279832	-0.003622
neighbors_att	0.565568	0.004651	-0.005775	0.008496	-0.008483	-0.025766	0.025766	0.567791	0.028240	-0.182727	-0.292075	0.604342	-0.060123	-0.196176	-0.242751	1.000000	0.591731	0.195908
Maxspeed	0.432753	0.006091	-0.004085	0.000539	-0.003223	-0.041826	0.041826	0.705619	-0.075904	-0.016328	-0.399843	0.516978	-0.070917	-0.253631	-0.279832	0.591731	1.000000	0.090405
school	0.086047	-0.009745	0.002042	0.013221	-0.005467	0.011953	-0.011953	-0.032497	0.080569	-0.016689	-0.029774	0.082356	0.212926	0.028364	-0.003622	0.195908	0.090405	1.000000

Figure 12: Correlation between features

Glossary

DNN deep neural network. 7

ETA estimated time of arrival. 2

FCD floating car data. 4

GIS Geographic Information System. 2

References

[1] M. Bai, Y. Lin, M. Ma, and P. Wang, "Travel-time prediction methods: A review," in *Smart Computing and Communication*, M. Qiu, Ed., Cham: Springer International Publishing, 2018, pp. 67–77, ISBN: 978-3-030-05755-8.

[2] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18, London, United Kingdom: ACM, 2018, pp. 858–866, ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3219900. [Online]. Available: http://doi.acm.org/10.1145/3219819.3219900.

[3] J. Dong and H. Mahmassani, "Stochastic modeling of traffic flow breakdown phenomenon: Application to predicting travel time reliability," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, pp. 1803–1809, Dec. 2012. DOI: 10.1109/TITS.2012.2207433.

[4] A. Skabardonis and N. Geroliminis, "Real-time estimation of travel times on signalized arterials," Tech. Rep., 2005.

[5] X. Zhang and J. A. Rice, "Short-term travel time prediction," *Transportation Research Part C: Emerging Technologies*, vol. 11, no. 3-4, pp. 187–210, 2003.

[6] J. Sun, C. Zhang, S. Chen, R. Xue, and Z. Peng, "Route travel time estimation based on seasonal model and kalman filtering algorithm," *J. Chang. Univ*, vol. 34, pp. 145–151, 2014.

[7] N. Wisitpongphan, W. Jitsakul, and D. Jieamumporn, "Travel time prediction using multi-layer feed forward artificial neural network," in *2012 Fourth International Conference on Computational Intelligence, Communication Systems and Networks*, IEEE, 2012, pp. 326–330.

[8] S.-Y. Yun, S. Namkoong, J.-H. Rho, S.-W. Shin, and J.-U. Choi, "A performance evaluation of neural network models in traffic volume forecasting," *Mathematical and Computer Modelling*, vol. 27, no. 9-11, pp. 293–310, 1998.

[9] R. Noland and J. Polak, "Travel time variability: A review of theoretical and empirical issues," *Transport Reviews*, vol. 22, pp. 39–54, Jan. 2002. DOI: 10.1080/01441640010022456.

[10] M. Elkær Holm, L. Vinther Kirkeby, and M. Jarlund, *Estimating travel time probability distributions using a mixture density network.*

[11] C. M. Bishop, "Mixture density networks," Tech. Rep., 1994.

[12] I. Guyon and A. Elisseeff, "An introduction of variable and feature selection," *J. Machine Learning Research Special Issue on Variable and Feature Selection*, vol. 3, pp. 1157–1182, Jan. 2003. DOI: 10.1162/153244303322753616.

[13] S. Kaufman, S. Rosset, and C. Perlich, "Leakage in data mining: Formulation, detection, and avoidance," vol. 6, Jan. 2011, pp. 556–563. DOI: 10.1145/2020408.2020496.

[15] osm wiki, *Map features*, 2019. [Online]. Available: https://wiki.openstreetmap.org/wiki/Map_Features.

[16] tomtom, *Copenhagen traffic*, 2020. [Online]. Available: https://www.tomtom.com/en_gb/traffic-index/copenhagen-traffic/.

[17] Tobias Skovgaard Jepsen, Christian S. Jensen, Thomas Dyhre Nielsen, Kristian Torp, *On network embedding formachine learning on road networks:a case study on the danish road network*, 2019. [Online]. Available: https://arxiv.org/pdf/1911.06217.pdf.