Using Spatial and Temporal Context for Predicting Energy Consumption of Electric Vehicles

Jacob Hjort Bundgaard Bjarke Bak Toxværd Madsen
Christoffer Popp Nørskov

June 11, 2019

Summary

Electric vehicles are becoming increasingly common due to increased focus on climate change and greenhouse gas emissions. Despite increase in the popularity and range of electric vehicles, range anxiety remains a concern for owners. Due to this, it is important that electric vehicle owners can get accurate energy consumption predictions when planning a trip. For this purpose, we propose a number of machine learning models that utilize a data-driven approach to prediction of energy consumption of electric vehicles. The proposed models consider many complex interactions related to the contextual nature of the driving data.

We use a data set containing driving information from 176 electric vehicles for a total of 2,461,356.8 kilometers driven, covering 27% of all roads in Denmark. This data was map-matched to a road map of Denmark and subsequently processed further by us. We integrate a number of external data sources with our dataset, with the goal of improving prediction performance. The first of these data sources is a high-resolution height map of Denmark, used to annotate the road network with inclines. The second data source is a speedmap obtained from Vejdirektoratet, which indicates average speeds for given time periods for certain roads in Denmark. The final data source is additional and extracted data from the OpenStreetMap (OSM) dataset, used to annotate features describing roundabouts, traffic lights, turning angles, and the number of road segments connected at each end of a given road segment.

This data is tested on the four proposed models: our baseline, a DNN which doesn't look at context; our supersegment model, a DNN which exploits context; an RNN; and an extended RNN (ERNN). The architecture of the ERNN includes a method of embedding categorical features as real feature vectors, which provides insights into the energy consumption patterns that the model learns for these features. We find that the ERNN architecture outperforms the other models, and achieves a mean absolute percentage error on trips of 18.77%. The results are in line with other work in this area, suggesting that the proposed ERNN architecture sufficiently captures many of the intricacies of energy consumption of electric vehicles in our dataset.

AALBORG UNIVERSITY

STUDENT REPORT

Title:
Using Spatial and Temporal Context for Predicting Energy Consumption of Electric Vehicles

Project period:
2019/02/01 - 2019/06/11
10th Semester of Computer Science

Project group:
mi105f19

Participants:
Jacob Hjort Bundgaard
Bjarke Bak Toxværd Madsen
Christoffer Popp Nørskov

Supervisors:
Thomas Dyhre Nielsen and
Kristian Torp

Abstract: The number of electric vehicles in use worldwide is rising every year. Even with improvements in battery technology giving longer range to electric vehicles, many owners still worry about the range of their vehicles. This worry is termed *range anxiety*. Obtaining more precise energy predictions can combat range anxiety, as they allow the owners of electric vehicles to more precisely gauge the range of their vehicles.
In this report, we detail the creation of multiple machine learning models that predict energy consumption of electric vehicles by exploiting contextual information related to trips conducted by these vehicles.
We base this project on a large dataset of electric vehicle driving data, which describes trips on the Danish road network. A number of external data sources are integrated with the dataset, such as height data obtained from a height raster of Denmark, and a speedmap from Vejdirektoratet.
We conclude that a Recurrent Neural Network architecture is able to successfully exploit the sequential nature of the context data to improve significantly upon a baseline model.

Number of pages: 68
Number of appendices: 7
Completed on: June 11, 2019

Contents

Preface

This report follows in the footsteps of Bundgaard et al. [7], which we co-authored. We use the same dataset, with the processed and newly introduced attributes that follow, as the basis for this report. Additionally, we use the best performing model that resulted from the report as the baseline for comparison in this report.

Chapter 1 borrows from the previous report, as this information is still very relevant to include for comprehension purposes. It is partially rewritten to focus on the new goal of exploiting context information available for trips.

Chapter 2 has undergone revision, with a new focus on work that proposes models which exploit sequential data formats within traffic related areas.

Chapter 3 borrows from the explanation of the original dataset, but otherwise includes original integration and analysis of new data.

Chapter 4 borrows the node2vec section from our previous work, but is otherwise focused on our proposed models that exploit context.

Chapter 5 and Chapter 6 are completely new and original additions to this project.

Glossary

- Road segment:
  - A stretch of road in the road network.
- Trip segment:
  - An instance of a vehicle traveling over a road segment.
- Trip / Trajectory:
  - A collection of trip segments that make up a trip.
- OpenStreetMap / OSM:
  - The provider of the underlying map of road segments that is used in this project.
- Geometry / Geography:
  - A collection of latitude/longitude coordinates describing lines or points in a spatial system. See Section 3.1.
- GPS point:
  - A point at X,Y that is collected as part of a trip. They are map-matched to road segments to form trip segments.
- Embedding:
  - A dense feature vector in $\mathbb{R}^m$ where m is the embedding dimension.

Chapter 1

Introduction

Electric vehicles are becoming a larger and more important part of private and commercial vehicle fleets. In 2017, there were more than three million electric and hybrid vehicles in use worldwide [18]. This trend is driven by concerns over the environmental effects of conventional vehicles, by policies promoting environmentally friendly vehicles, and by advantages to the driving characteristics and recurring costs of electric vehicles [37].

The demand for battery electric vehicles (from here on electric vehicles) has led to improvements in battery technology and electric vehicle manufacturing. In 2011, a typical electric vehicle had a range of 160 km [24], while newer vehicles have advertised ranges up to 530 km [29, 17, 34]. Despite this, some customers are still reluctant to switch from conventional vehicles to electric vehicles, since the real-world range is still lower than for conventional vehicles and the batteries take longer time to charge than refuelling conventional vehicles.

Many electric vehicle owners report being worried about if they will be able to drive to their destination without running out of battery or having to charge [27]. This phenomenon has been termed *range anxiety*. Range anxiety is exaggerated by changes in energy consumption during different seasons of the year, and for different traffic conditions, which makes it hard for drivers to predict the range of their vehicle, even for trips they have taken before [22].

It is therefore of benefit to electric vehicle drivers to get more accurate predictions of energy consumption for trips, taking into account factors which may affect it, such as the road layout, air temperature and other environmental factors.

Other factors that may affect the energy consumption are those related to the context of the trips. For example, when driving on a motorway, the energy consumption of an electric vehicle may be influenced by the context of whether the driver recently drove onto the motorway via an on-ramp and is therefore accelerating.

Alternatively, the driver may have been driving on the motorway for a while and is therefore keeping a constant speed. As such, models for electric vehicle energy prediction may benefit from incorporating this type of context.

We use a dataset containing driving information from 176 electric vehicles for a total of 2,461,356.8 kilometers driven, covering 27% of all roads in Denmark. This data was map-matched to a road map of Denmark and subsequently processed further by us.

Given this dataset, with the goal of obtaining accurate energy predictions for it by exploiting the sequential nature of trip data, we define the following problem statement:

- Is it possible to accurately predict energy consumption for electric vehicles given the complex non-linear interactions between the factors in our dataset by using neural networks?

– Is it possible to exploit the sequential nature of trip data to improve energy predictions?
– Can data about the intersections a trip goes through improve the energy predictions?
– How does different handling of categorical features influence the energy prediction?
– How can the prediction model help understand underlying patterns in the data?

Chapter 2

Related Work

2.1 Energy Prediction

Varga et al. [38] perform a review of current works within range and energy prediction for electric vehicles.

They highlight that range anxiety is a real concern for many new electric vehicle drivers, leading them to use only 80% of the actual range of their car, and that experienced drivers have lower range anxiety. This motivates the aspiration to improve energy consumption and range predictions.

They determine that many factors influence energy consumption, including trip patterns, traffic flow, and environmental factors. Among the environmental factors of temperature, precipitation, wind, etc., they show that temperature has the most significant influence on range, with the cooling load in summer causing a 17.2-37.1% range reduction and the heating load in winter causing a 17.1-54.0% range reduction. They highlight experimental range reductions from 150 km at ambient temperatures of 20°C to 85 km at 0°C and 50 km at -15°C. This aligns with previous experience that temperature and associated features, such as seasons, have a measurable influence on the performance of models that predict energy consumption[7], and motivates the further inclusion of these features.

They also highlight a number of factors for the energy consumption which we do not have the data to consider in our model. One of these factors is the passenger load of the vehicle, which can increase the total weight of the car by 15-25%. Another is the tire pressure, which at optimal levels can reduce energy consumption by 10%. These factors put a natural limit on the accuracy of a model that doesn't consider them. To handle multiple factors simultaneously, they highlight that an approach that splits the road network into discrete road segments, represented as a graph, can be used successfully to assign specific attributes such as traffic intensity, air temperature, and incline. This method for representing a road network as a graph is often used, as it lends itself well for calculating efficient paths based on a cost associated with traversing an edge [5, 6, 3]. Approaches that do not use such a discretization process, such as the one proposed by Zheng et al. [45] for range estimation of electric vehicles, have a significant downside in that time-series of GPS points with related information cannot be known in advance. This limits the model's capability for range estimation, even if future planned routes are known.

Finally, they note that many studies look at single factors for electric vehicle range and recommend continued research on data-driven methods that take into account the many factors that influence energy consumption and range.

Bundgaard et al. [7] created a regression model for predicting the energy consumption of

electric vehicles over trip segments, which are the parts of trips that model traversing individual road segments. They used the graph approach noted by Varga et al. [38] and found that an embedding of the road segments of the road network as a dense feature vector significantly contributes to the performance of a segment-based model for predicting energy consumption. Both node2vec[14] and GraphSAGE[15] were used to generate these embeddings of the road network to include spatial information in their model.

Additionally, incorporating information from more than one segment at once, due to the fact that energy consumption over one segment is dependent on the segment immediately before or afterwards, is highlighted as a future work. A way of illustrating this is if there is a change in road condition, for example when driving onto a motorway from a residential road which necessitates acceleration to motorway speeds. The model created by Bundgaard et al. [7] modeled the interactions between features that describes one trip segment, and the corresponding energy consumption over that segment. We aim to improve upon this work by creating an energy prediction model which can exploit this contextual information.

De Cauwer et al. [9, 10] also create an energy prediction model for predicting energy consumption over individual segments. They employ a data-driven approach using two datasets, with the largest dataset consisting of trips driven on a mix of highway, rural and urban roads over a period of over one year by 30 vehicles, and the other primarily within a specific urban area by three vehicles. De Cauwer et al. [10] state that only a representative subset of the data is linked with altitude and road information. The resulting data after filtering and linking with external sources concerns 3700 km driven by three of the 30 vehicles from the first dataset, and 10,700 km driven by two vehicles from the second dataset. Their energy prediction approach is based on the physical relationship between the work that a vehicle performs to move, under the influence of external factors, and the resulting energy consumption of the electric vehicle over a road segment. They model the energy consumption over segments by using a neural network to predict a speed factor and a motion factor that describe the accelerations and decelerations performed over a segment. These predictions are used as input to a Multiple Linear Regression model, in addition to other external features related to the trip such as ambient temperature or road category of a given segment. However, their approach exclusively considers information from individual segments and therefore does not learn cross-segment interactions.

2.2 Fuel Consumption Estimation

Fuel consumption estimation is the conventional vehicle pendant of energy prediction. Although the car powertrain is different, the fuel consumption of conventional vehicles depends on many of the same factors as the energy consumption of electric vehicles.

Kanarachos et al. [20] create a model that tries to predict instantaneous fuel consumption using smartphone GPS position and altitude, speed, and triaxial acceleration as inputs. They compare Long-Short-Term-Memory (LSTM) Neural Networks, tuning the number of hidden units from 100 to 500 and training with stochastic gradient descent and the Adam optimizer, and a variant of Recurrent Neural Networks called NARX networks, trained using the Levenberg-Marquardt method. They show that RNNs, both the LSTM and the NARX variants, can be beneficial for instantaneous fuel consumption estimation. RNNs are also used for problems more closely related to ours, such as for travel time estimation.

2.3 Travel Time Estimation

Travel time estimation is closely related to energy prediction, in that it also tries to predict an aggregated attribute of trips over a road network. One model for travel time estimation is proposed by Zhang et al. [43]. They divide the geographical area for the model, for example a city such as Shanghai, into a grid with cells of a fixed size. Then, rather than model a trip as individual trip segments on roads, as [10, 7], they map the GPS points of a trajectory onto this grid, then model trips as a sequence of grid cells that have been driven through. Zhang et al. [43] also include a number of categorical features that describe the temporal component of their data. They represent these discrete temporal features by learning embeddings for them. Additionally, they add spatial feature vectors for their grid cells, but do not describe the method for learning these vectors. They then estimate the total travel time of trips by use of a recurrent neural network. As such, Zhang et al. [43] exploits the context provided by the trip sequences for time travel estimation. They show that, by considering more of this context, they improve significantly on their predictions. We don't consider the grid-based nature of the model by Zhang et al. [43] to be key to their contribution. They highlight that previous segment-based models did not capture cross-segment interactions, but do not explain why capturing these interactions are only possible with a grid-based model. On the contrary, we find that applying context to a segment-based dataset is just as possible as for a grid-based dataset, and that it allows for many additional features, such as embeddings of the road network, road categories and angles between segments. We also have a significant number of categorical attributes for our trip segments, and creating embeddings for these is likely to improve the quality of the energy predictions, as Zhang et al. [43] found that their spatio-temporal embeddings had a positive contribution to their model's predictive performance.

Another traffic-related approach that uses RNNs is proposed by Wu et al. [40], which mines trajectory (trip-related) information from sequences of trips, matched to road segments. Inspired by the successful use of models based on recurrent neural networks in other traffic-related areas, we aim to use RNNs for sequence-based learning for obtaining energy consumption predictions for trip segments. To the best of our knowledge, no such attempt to include a larger context than individual segments has been done for energy prediction. As it is possible to model trips as a sequence of trip segments in our dataset, it is indeed relevant to consider this sequential thinking for our energy prediction model to determine the usefulness of more context.

Chapter 3

Data

In this chapter, we explain the data foundation of this project, as well as additions we make to this data in the form of integration of height data, including additional information about the road network and merging information from an external speedmap. We perform an analysis of the attributes available in our dataset, to determine their correlation with the energy consumption in our electric vehicle driving data.

3.1 Data Basis

The data used for this project is a result of the work done by Andersen et al. [1, 2]. It consists of trip information of electric vehicles driving in Denmark in the period of 2012-2014, collected into a single database using PostgreSQL 9.6.10 with PostGIS 2.3.3. The relevant structure of the database can be seen in Figure 3.1. `osm_table` defines the road network and contains road segment geometries with related information. `trip_table` defines the parts of trips, namely the GPS points contained in `gps_point_table`, that traverse specific road segments, and relates these to external information. The orange tables (`date_table` and `time_table`) contain temporal data such as month and weekday. The blue tables (`weather_table` and `weather_station_table`) contain data about weather stations and their measurements.

The map used by this project is a road map of Denmark from OpenStreetMap (OSM), version January 1st, 2014. This is the table, `osm_table`, in Figure 3.1. This map consists of 789,371 individual road segments, spanning a total of 134,730.72 kilometers.

Each road segment has a so-called *LineString* geography, which consists of a sequence of points, each specified by a latitude and a longitude, connected by straight lines. The coordinate system used for the points that make up our road segments is WGS84, also known as EPSG4326. As an example, Figure 3.2 shows a road segment consisting of five points (shown as red dots). The segment has a 'direction' attribute which indicates whether you can drive only 'forward' or in 'both' directions on the segment. Additionally, the speedlimit_forward and speedlimit_backward attributes from the `osm_table` are very sparse, so we use the values that are available to calculate an average speedlimit for each road category. When driving in the forward direction on the road segment, the user drives along the lines in the direction indicated by the order of the points in the sequence (shown as blue arrows). The segment is represented textually as:

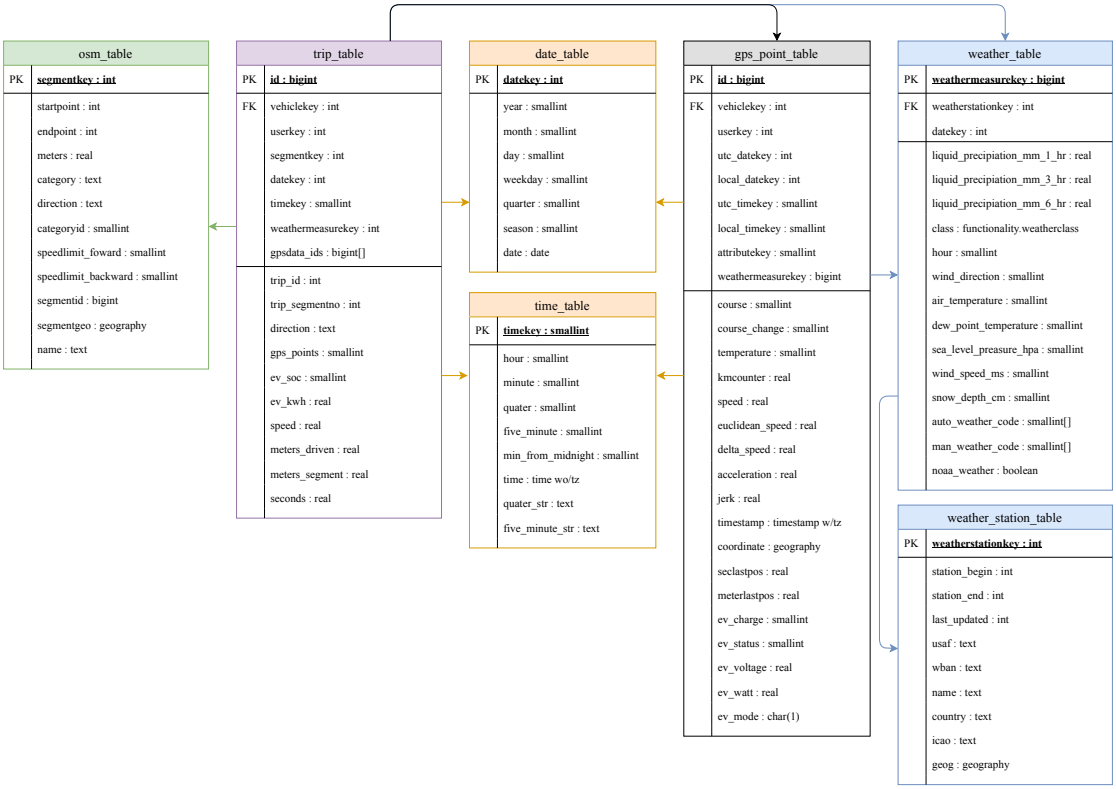


Figure 3.1: Database Structure

```
LINESTRING(12.0721567 55.6341933, 12.0722725 55.6346091,
           12.071761  55.6346732, 12.0711819 55.6347177,
           12.0706047 55.6346395)
```

An additional attribute we add to the OSM data is the degree of each startpoint and endpoint of a road segment. This is the number of other segments that connect to that point of the segment.

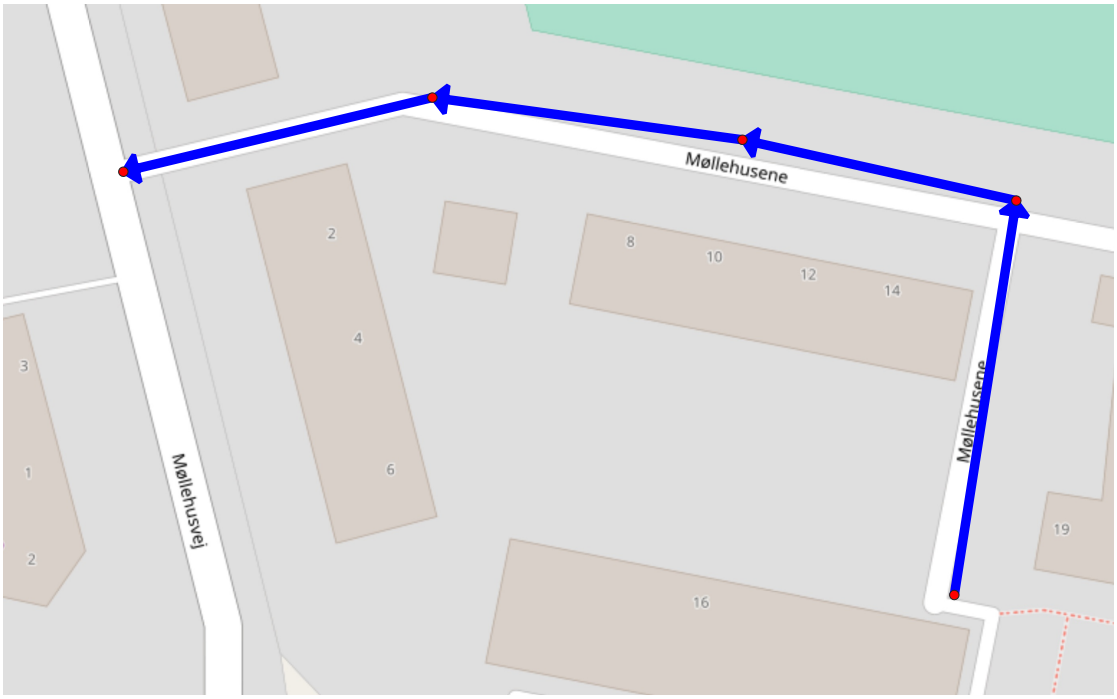


Figure 3.2: One OSM road segment consisting of five points.

Figure 3.2 additionally illustrates the use of OSM as a background for map figures in this report. The background is also based on the OSM dataset, but on a newer version and rendered in a different way than the elements drawn on top of it. This can lead to slight offsets between the foreground and background, as shown in the figure. In this case, the points, lines and arrows shown in the foreground should be regarded as the truth, while the background is purely there to provide context.

This map was used for map-matching driving data collected between 2012 and 2014 by the Danish Energy Agency and the company Clever, as part of the *Test an electric vehicle* program run by Clever [35]. This program involved lending out 176 electric vehicles of the models Mitsubishi i-MiEV, Peugeot iON, and Citroën C-Zero. These car models are effectively identical, as they are all built on Mitsubishi's i-MiEV platform. For the experiment, each vehicle was used in three month cycles, after which the vehicle would be reassigned to another person.

The raw data collected from this program consists of 218,834,510 GPS points (`gps_point_table`) sampled at a rate of one Hz and annotated with CAN bus readings of several attributes such as speed, energy consumption in Watts, state of charge, system time, and compass heading. This data has been cleaned and processed by Krogh et al. [22, 1, 2]. As part of this processing, the trip data was map-matched to OSM by using a Hidden Markov Model based map-matching algorithm [22, 28].

This map-matching consists of mapping the GPS points onto the road segments in OSM, and grouping consecutive GPS points into trip segments based on the road segment they are mapped to. This is the `trip_table` as seen in Figure 3.1. There is a total of 14,473,006 trip segments. Each trip segment is associated with the road segment that it was driven on, and is annotated with the information shown in the corresponding table in Figure 3.1. This information was extracted from the GPS points that were grouped into that trip segment. Consecutive trip segments are then further grouped into complete trips (or trajectories). A new trip is created when the time difference between two consecutive location updates exceeded three minutes [22]. This process resulted in 275,994 trips, representing a total of 2,461,356.8 kilometers driven. In Figure 3.3 we show the complete coverage of this driving data over a map of Denmark. Grey lines indicate the presence of few trips, and black lines indicate more trips conducted in those areas.

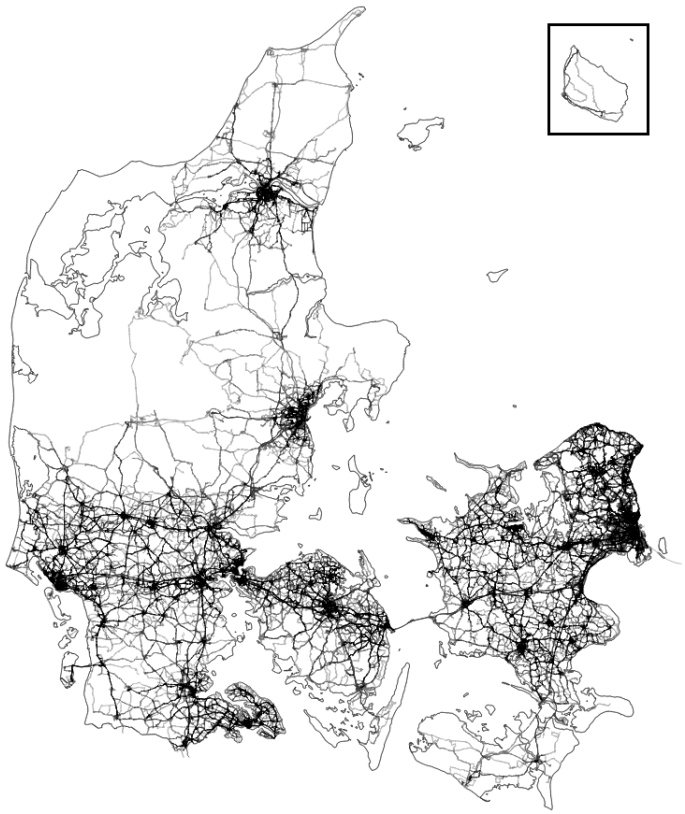


Figure 3.3: Coverage of Denmark for the driving data.

For attributes related to the weather, historical data collected from weather stations around Denmark is used. See Figure 3.4 for the locations of the weather stations. This weather data was published by Danmarks Meteorologiske Institut (DMI) through the National Oceanic and Atmospheric Administration (NOAA) and was integrated in the dataset by Krogh et al. [22]. The weather data is used to further annotate trip segments with information about temperature and wind, to allow analysis on the effect of these factors on the energy consumption of the electric vehicles. Through an investigation of the weather data, we found that five days had missing

weather data. The data for these days is available from NOAA[26], allowing us to remedy this by re-including the data for the five days in our dataset. In order to capture the influence of headwind on a traveling vehicle, we isolate headwind as wind speed parallel to and in the opposite direction of the driving direction of the car, as described in Algorithm 1. Note that a negative headwind speed is interpreted as tailwind. This is the algorithm used by Bundgaard et al. [7], with the following notes: We assume that angles automatically wrap around when crossing 0° or 360°. `ST_Azimuth` is used to calculate the angle of a line, defined by the geometries of two points, clockwise from north. As such, `car_direction` is the direction that the vehicle is traveling *to*, and `wind_direction` is the direction from which the wind blows *from*.

- $t.road\_segment$ refers to the road segment geometry for a trip segment t.
- *start* is the startpoint of a road segment.
- *end* is the endpoint of a road segment.

Algorithm 1 Calculates headwind speed from the wind data. Adapted from [7].

Input: A set of trip segments T and their associated road segment geometries.

```
function CalculateHeadwind(T)
    for all t ∈ T do
        road_direction ← ST_Azimuth(t.road_segment.start.geo, t.road_segment.end.geo)
        if t.direction = "BACKWARD" then
            car_direction ← road_direction − 180°
        else if t.direction = "FORWARD" then
            car_direction ← road_direction
        end if
        if t.wind_direction − car_direction > 180 then
            relative_direction ← t.wind_direction − car_direction − 180°
        else if t.wind_direction − car_direction < 180 then
            relative_direction ← t.wind_direction − car_direction + 180°
        end if
        t.headwind_speed ← −t.wind_speed ∗ cos(relative_direction)
    end for
end function
```

3.1.1 Data Analysis

We now present an analysis of the correlation between the speed vehicles traverse segments at in our dataset, and the energy consumption of those vehicles. Following that, we give an analysis of the information derived from the `weather_table`, namely temperature and our calculated headwind speed. The final attributes available from Figure 3.1 are the ones from the map itself (category of the road) for *where* people are driving, as well as from the `date_table` and the `time_table`, that include the temporal elements of our data for *when* people are driving.

Determining the impact that these factors have on the energy consumption of the vehicles, can guide our decision in including these in an eventual energy prediction model and determine why they contribute or do not contribute to the predictive performance of said model.

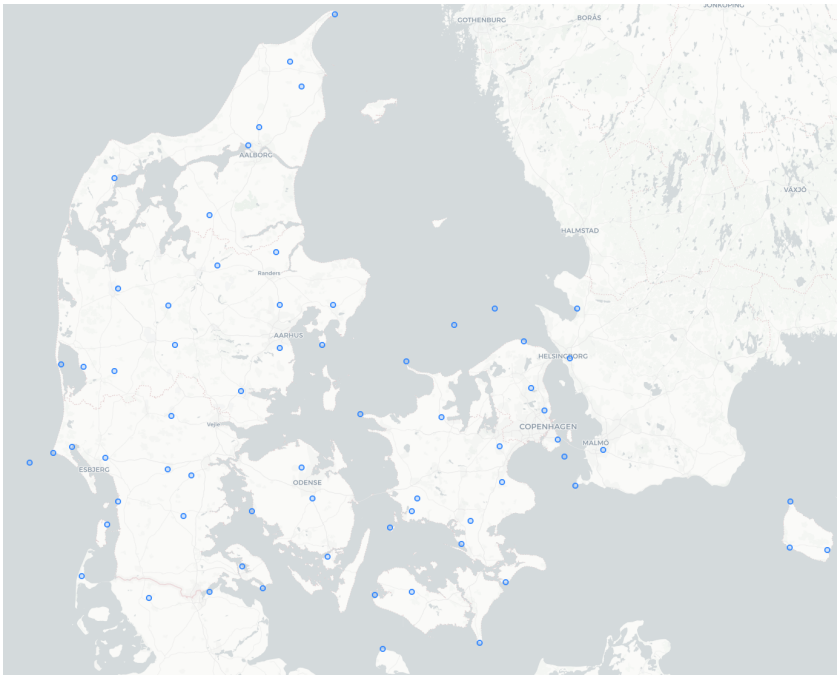


Figure 3.4: Location of weather stations providing weather data for the trip segments.

Speed

The speed at which vehicles traverse a segment is likely to have a significant impact on the energy consumption. Our analysis of the correlation between speed and average energy consumption can be seen in Figure 3.5. Here, the blue line indicates the average energy consumption at a given speed, grouped in 1 km/h increments. We exclude speeds above 130 km/h, as this is the top speed indicated for vehicles based on the I-MiEV platform[24]. This shows a non-linear correlation between the speed at which the vehicles drive, and the energy consumption required to move the vehicle. There is increased average energy consumption at speeds around 20 to 40 km/h, likely caused by acceleration to higher speeds or the presence of other traffic. From 40 to 60 km/h, the average energy consumption reduces, before increasing again after 60 km/h, where there is a trend of increased energy consumption as the speed increases. This conforms to our expectation that as vehicles drive faster, more energy has to be expended to combat external forces such as wind resistance[25].

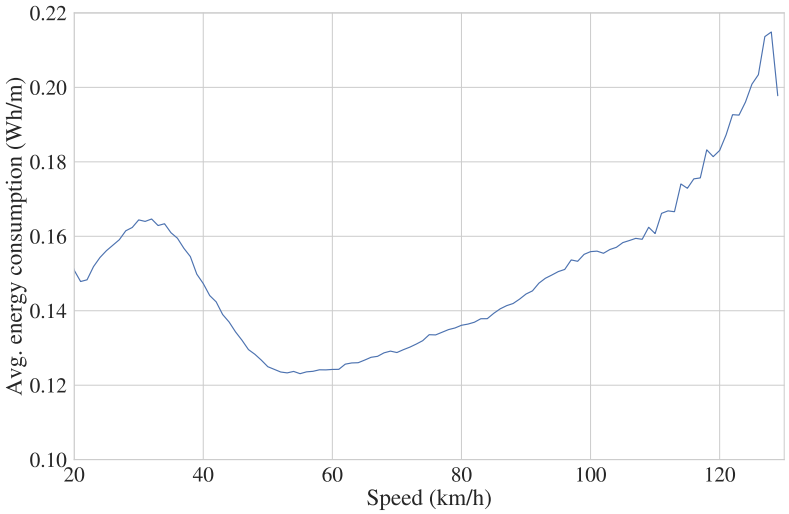


Figure 3.5: Impact of speed of a vehicle on the energy consumption.

Temperature

Temperature is known to have a significant impact on the efficiency of electric vehicles. A lower temperature means that more energy has to be used by the auxiliary systems to heat the cabin as well as the battery[42]. This is also reflected in our analysis of the correlation between temperature and energy consumption, which can be seen in Figure 3.6. It is evident that the optimal temperature to drive at is centered around 20°C. Reducing the temperature seems to, on average, increase the average energy consumed per meter by the electric vehicles significantly. The average energy consumed per meter by an electric vehicle driving in 20°C weather costs on average half the energy versus driving in -10°C weather. This conforms with the results from Yuksel and Michalek [42], that colder weather reduces the efficiency of electric vehicles due to increased need for heating, leading to reduced efficiency.

Wind

To determine the correlation between our wind data and the energy consumption data, we analyze the effect of headwind as calculated with Algorithm 1. We expect that headwind significantly effects energy consumption, as vehicles are designed with aerodynamics in mind to reduce fuel and energy consumption at high speeds. The analysis can be seen in Figure 3.7. Headwind magnitude below -5 (meaning 5 m/s tailwind) seems to have a weak correlation with energy consumption, likely due to other factors influencing the energy consumption or errors in the weather data. Above -5 m/s, there seems to be a trend with reduced energy consumption as the amount of tailwind increases. When there is a headwind (above 0 m/s), there is a weak increase in average energy consumption as the headwind magnitude increases. The change in energy consumption does seem to correlate with changes in headwind speeds, however not as clearly as with temperature. This is likely because wind is more local to the weather stations than temperature, due to buildings or other topography that we are not able to account for with our data.

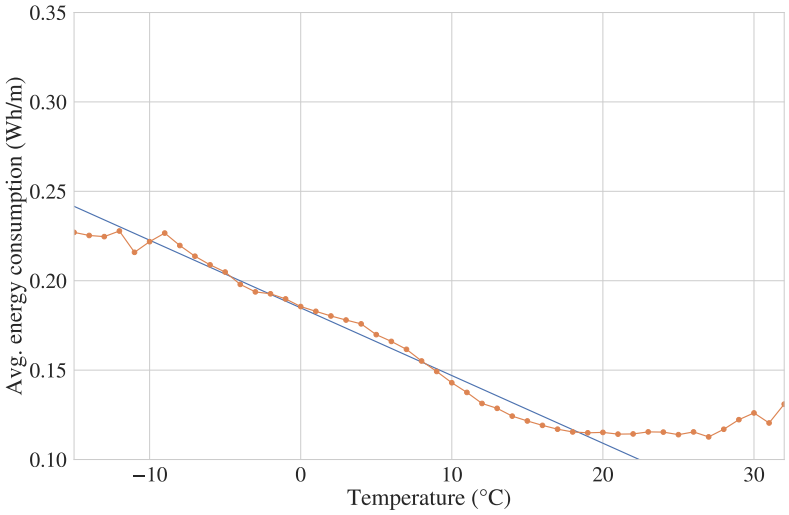


Figure 3.6: Impact of temperature on the energy consumption.

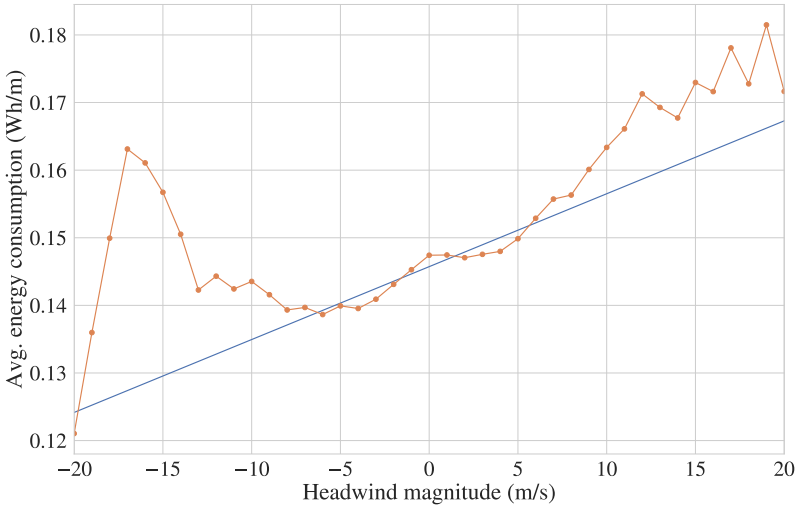


Figure 3.7: Impact of wind speed on the energy consumption.

Month

We include an analysis of the month attribute, though we expect it to primarily be an indicator of temperature. Our analysis of temperature showed a direct correlation between temperature and energy consumption, with lower temperatures resulting in a higher average energy consumption. We expect the same for the various months, where the winter months have a higher energy consumption compared to the summer months. The change in energy consumption per month can be seen in Figure 3.8. A clear distinction between the seasons is evident, as the fall/winter months have a significantly higher average energy consumption compared to the spring/summer months. This conforms with our expectation that month is an indicator of temperature, as it follows the logic that the colder months have a higher average energy consumption.

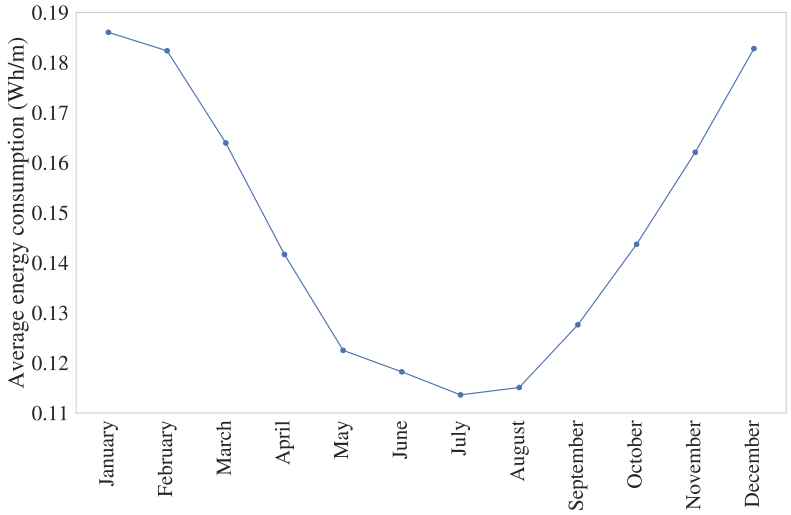


Figure 3.8: Change in energy consumption per month.

Weekday & Time of Day

The weekday attribute tells us which day a trip is driven. Our hypothesis is that the day is not a strong indicator of change in energy consumption. A week can be split into workdays and weekends, and we expect that there is a different correlation between rush hour and energy consumption in the weekends versus the workdays. To test this, we first calculate an average energy consumption per weekday, which can be seen in Figure 3.9. There is almost no change in average energy consumption per meter driven over the complete day, whether it be a workday or weekend. In Figure 3.10, we analyze the change in energy consumption per hour for workdays and saturday/sunday. We see that as the day progresses, there is little difference between saturday and sunday. There is a drop in average energy consumption after 7:00 during the workdays, compared to the weekend days, likely attributed to the fact that people do not drive when they are working. Sunday seems to stand out from the other days, as the early morning bump in energy consumption around 7:00 is not present. The increase in average energy consumption during the evening could be attributed to the fact that it is colder during evenings, leading to reductions in the efficiency of the vehicles. This means that the weekday may be an indicator of change in energy consumption when paired with the time of day feature, even if it by itself does

not indicate large changes in average energy consumption.

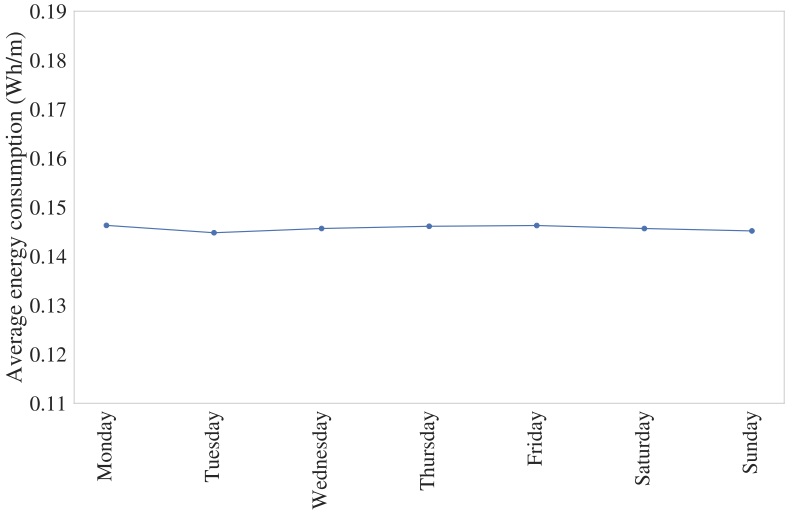


Figure 3.9: Change in energy consumption per weekday.

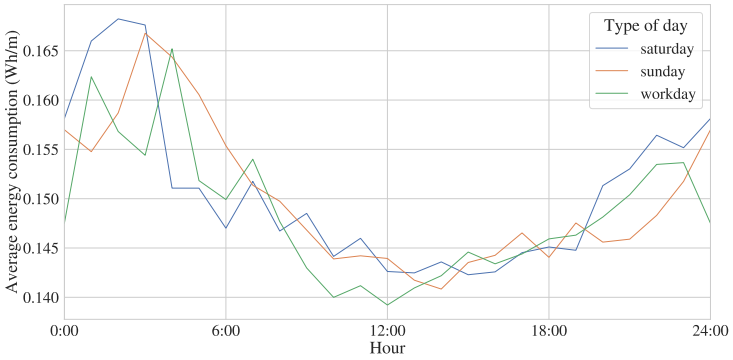


Figure 3.10: Change in energy consumption over time for workdays and weekend days.

Category & Time of Day

The category of the road that people drive on indirectly affects speed patterns, with people driving differently in residential areas vs. on motorways. This is because each of the categories have different speed limits in Denmark, leading to changes in speed for each category. The various road categories are therefore likely to correlate with energy based on the speed that people drive on them. We expect that categories for major arteries such as motorways have differing energy consumption patterns from categories such as residential roads. In Figure 3.11 we show an analysis of the average energy consumption patterns over the course of 24 hours, for each of the top five most-driven road categories. Note that the data is for weekdays only, and

the spikes prior to 6:00 is due to limited amounts of data, as few people drive at these times in our dataset. The figure shows that, on average, vehicles use more energy when driving on residential roads compared to motorways or the other major arteries. A clear distinction can also be made between the average energy consumption in the early hours during rush hour, from around 6:00 to 8:00, and afterwards. This is likely caused by rush hour traffic, which results in increased energy consumption. Following the morning traffic, it seems that each of the top 5 categories follow a similar trend of increased average energy consumption as people go home from work at 16:00 and afterwards. The increase in average energy consumption during the evening for all categories could, as was noted for the weekday analysis, be attributed to the fact that temperature decreases during the day, leading to reductions in the efficiency of the vehicles.

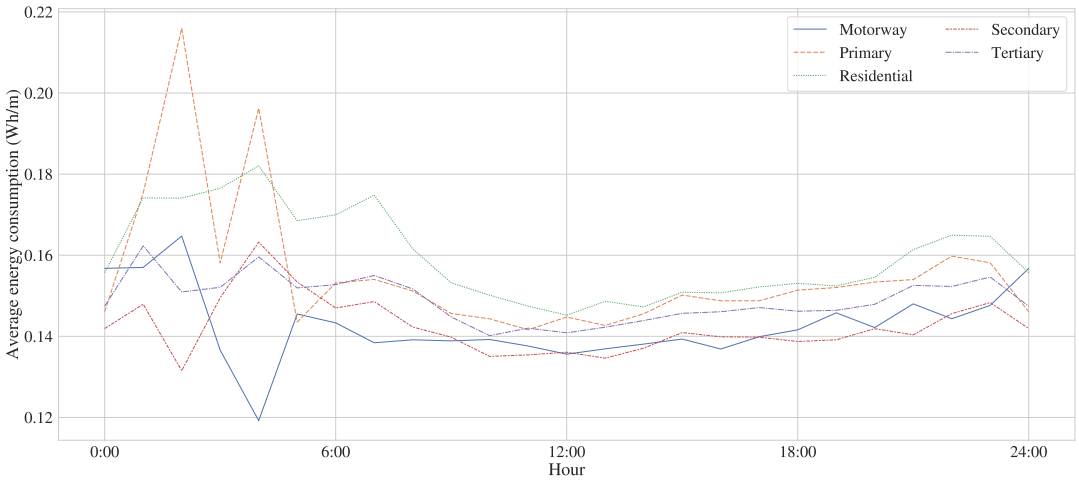


Figure 3.11: Change in energy consumption over time for the five most-driven road categories.

In the following sections, we describe our own augmentation of the dataset, adding new attributes and improving the quality of some existing attributes.

3.2 Height Data

The incline of a road segment is useful for determining the amount of work an electric vehicle has to perform to traverse the road segment. Driving on a stretch of road that has a steep incline requires more work than driving downhill. For electric vehicles this is further complicated by the use of regenerative braking, that can in some cases result in generation of energy on downhill stretches.

In Figure 3.12, we show the correlation between incline and how the average energy consumption develops with changes in incline. The yellow line indicates the average energy consumption at various incline levels. Please note that the regression is calculated for the raw data, and that the energy consumption is averaged in groups of 1% incline increments. From this analysis, we can see that there is a trend that, on average, as incline increases or decreases, the energy consumption will follow. The incline of a segment is therefore a good indicator of changes in energy consumption.

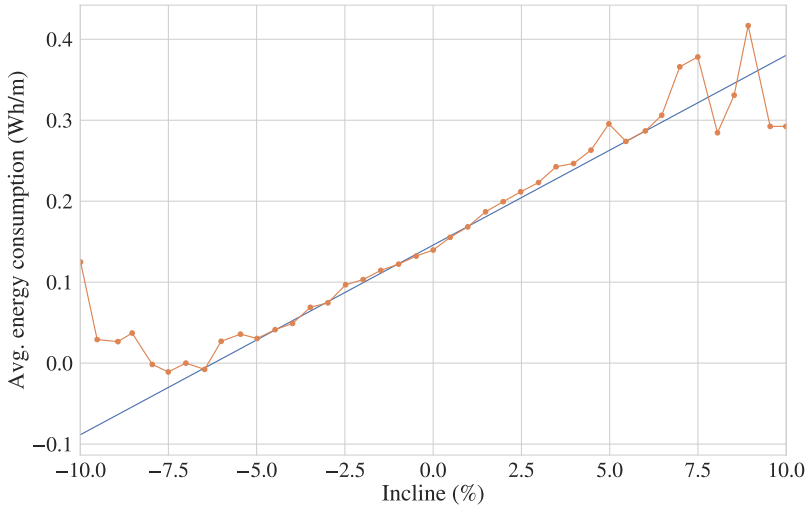


Figure 3.12: Impact of incline on the energy consumption.

To obtain this height data, we extend upon previous work which extended our electric vehicle dataset with inclines[7], calculated based on information from a height raster that was obtained from Kortforsyningen[21]. This raster consists of several tiles, which were reprojected to the coordinate system used by the OSM data in the original dataset. From this work, it was later found that the reprojected rasters were misaligned, leading to two meter gaps between the raster tiles with no height values present. This resulted in missing inclines for about 1.5% of all road segments. In this section, we describe how to obtain inclines for all road segments with this raster.

We found that the misalignment of raster tiles described by Bundgaard et al. [7] happens when individual raster tiles are reprojected to the destination coordinate system. To alleviate this issue, it is therefore necessary to first merge the raster tiles from Kortforsyningen to one raster, and then reproject the merged raster to the coordinate system used by OSM. We merge several raster tiles to a single raster by use of the GDAL library, which is available through the QGIS software[12, 33]. The merged raster is subsequently reprojected to the correct coordinate system with QGIS. A downscaled version of the merged raster covering all of Denmark can be seen in Figure 3.13, with darker pixels indicating a higher altitude.

Following the reprojection of the merged raster, we wish to use this height map to calculate the incline of each of the road segments in our data. To calculate the incline for road segments, we first need to determine the altitude of each startpoint and endpoint for all road segments. To determine the altitude of a point on the raster, we use PostgreSQL with a Geographic Information System extension called PostGIS[32]. This allows us to create an SQL query for which the database will use bounding boxes to determine which raster the point intersects with, if multiple raster tiles are used to represent the height map. Once it has found which raster the point intersects with, it then loads the raster to find the value of the pixel in the raster the point intersects with by use of the ST_NearestValue function. This loading process is expensive for large raster tiles, and needs to be performed for each of the approximately 1.5 million points. As such, it is infeasible to work on the single merged raster.

Instead, we tile the merged raster, which has a resolution of 357,490x161,233 and a size of approximately 200 GB, into tiles with a resolution of 625x625 and a size of 1.5 MB before

uploading them to the database. This process of splitting the raster into smaller tiles allows for more efficient lookups of values in the database, since only a single of these smaller tiles need to be loaded into memory to fetch the height value of the point intersecting with the tile. This method is significantly more computationally efficient than the approach taken by Bundgaard et al. [7], so we choose to work with the full 1.6x1.6m resolution raster.

Once the altitude of all startpoints and endpoints of road segments are obtained from the height raster, we use them in combination with the length of the segments to calculate the incline as a percentage. 2366 road segments (0.32%) have inclines above or below 10 and -10% incline, respectively, calculated through this process. Some of these roads are split into segments exactly under a bridge, and one of their points are therefore falsely assigned the height on top of the bridge. In other cases it seems to be an error from the raster itself, with no obvious reason for the errors. We have therefore chosen to clamp all roads above 10% to 10% and below -10% incline, to -10%, as there is no apparent pattern to this behaviour. These clamping values are motivated by a report from Vejdirektoratet[39] about roads built in Denmark, stating that roads built in Denmark must have an average incline of 10% or less.

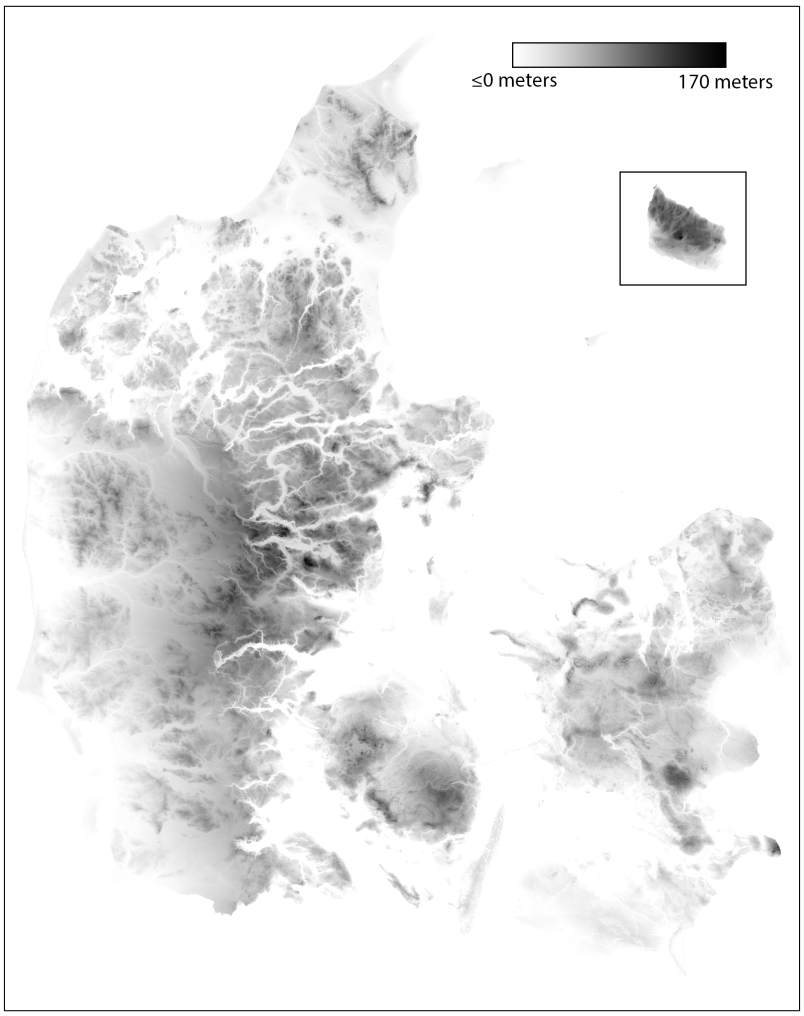


Figure 3.13: A resized version of the height map.

3.3 Additional OpenStreetMap Data

Bundgaard et al. [7] found that the addition of features that describe the road network itself, such as incline and the length of segments, improves the quality of the prediction model. De Cauwer et al. [10], in addition to using this type of data, include data describing traffic intersections and pedestrian crossings in their model for predicting speed-related factors for segments, suggesting that these types of features are also useful in determining the speed characteristics of a segment. Data about traffic intersections is tracked by OSM, however this is not included in the source data that was extracted from OSM, for our source data.

To combine this information with our source data, we use a publicly available archive of OSM called Geofabrik[13]. From this archive, we use a version of OSM Denmark from January 1st, 2014, matching the extract date of our source data. We then extract and map traffic intersection related attributes to our version of the OSM data. Of the attributes available in the OSM data extract, only the attributes related to traffic lights and roundabouts was sufficiently present to be useful, and as such only this information is described.

3.3.1 Traffic Lights

Intersections with traffic lights affect vehicle energy consumption, due to vehicles decelerating or accelerating as they enter or leave intersections with traffic lights[4, 36].

While traffic lights are often present in intersections, and the effect on energy consumption can therefore be partially described by turning angle, the traffic light itself is independent of the intersection, as it acts as a traffic flow control mechanism. Therefore, information about traffic lights works in conjunction with turning angle to describe intersections. This also allows us to describe the presence of traffic lights on road segments without intersections, such as pedestrian crossings in the middle of a road.

The difference in energy consumption, for our dataset, when driving up to and away from a traffic light can be seen in Figure 3.14. The figure contains box plots for energy consumption per meter when driving on road segments with traffic lights at either end, both ends, or at none of the ends. It shows that the average energy consumption is markedly higher when driving away from a traffic light than when driving up to one or driving on segments without traffic lights. Therefore, we can conclude that the presence of a traffic light at the start or end of a road segment is a strong indicator of a change in energy consumption of a vehicle driving on the segment. The data is limited to the road category on which traffic lights are most common, tertiary roads, to better allow comparison to driving on road segments without traffic lights at either end as this is also the most driven on category. Note that the `between or inside traffic lights` category includes segments inside of a single traffic light intersection, as well as segments between two different traffic lights, as there is no way for us to determine which of these groups a segment belongs to.

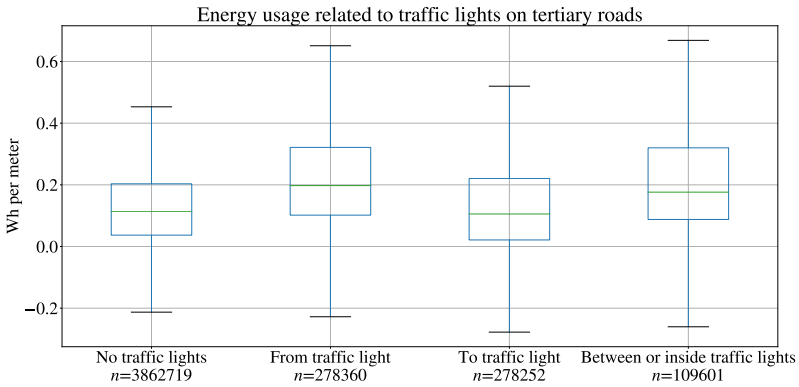


Figure 3.14: Energy consumption when driving up to, between, and away from traffic lights on tertiary roads.

Data about traffic lights is not originally provided in our dataset, and as such we include this ourselves. Using the traffic light information available from the OSM Denmark archive from Geofabrik [13], we add traffic light information to our dataset. The traffic light data from OSM consists of a series of points, each indicating the presence of a traffic light. To match these points to the segment in our map, we construct a buffer around each traffic light point. For each of these buffers, we determine which road segments intersect with it, and find the most common start- or endpoint among these, and mark that point with a traffic light attribute. For this attribute a value of 1 indicates the presence of a traffic light, and 0 otherwise. The size of the buffer for each traffic light must not be too large, as then segments which are not actually close to the traffic light will be marked as being part of it. Conversely, too small a buffer and segments that should be marked will not be. To determine the proper buffer size, we test buffer sizes up to 20 meters, and find that a buffer size of 20 meters provides a sufficient tradeoff between intersecting segments inside large intersections, and not intersecting unrelated segments outside smaller intersections.

A visualization of this approach can be seen in Figure 3.15. The red lines are road segments, and the red circle is the point with a radius of about 20 meters, indicating the presence of an intersection with traffic lights. For this specific intersection, it is also apparent that how OSM models the road network does not necessarily correspond one-to-one with how the roads appear in the real. This can be seen with the centermost line on the right in Figure 3.15, which models the two roads that merge to lead away from the intersection, as well as a single road that leads towards the intersection. SQL queries for obtaining this data can be seen in Appendix A.

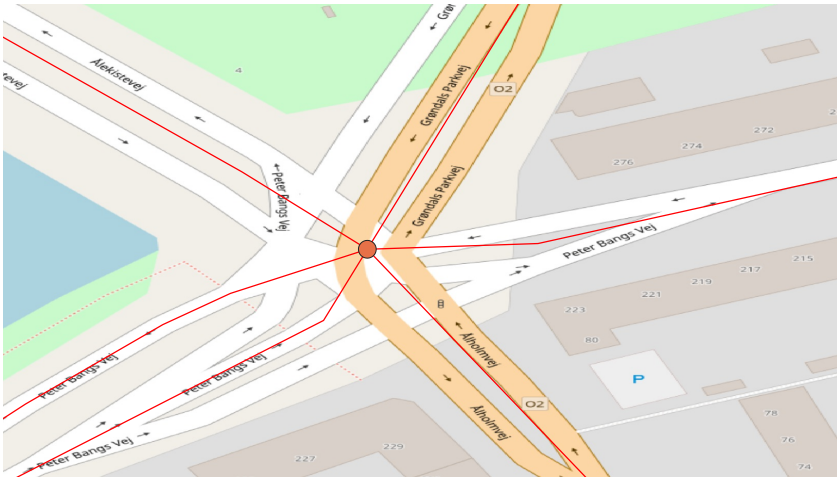


Figure 3.15: The red circle indicates the presence of a traffic light, with road segments intersecting it being marked with traffic light attributes.

3.3.2 Roundabouts

Roundabouts are likely to have a significant effect on the energy consumption of vehicles depending on whether the vehicles are entering, inside or leaving the roundabout. Traffic outside a roundabout must yield to traffic inside it, which means that a vehicle inside or leaving a roundabout will be likely to experience a higher energy consumption than a vehicle entering a roundabout. To verify this hypothesis, we perform an analysis of the effect that the presence of roundabouts have on the energy consumption of vehicles in our data.

The difference in energy consumption when driving up to, inside, and away from a roundabout can be seen in Figure 3.16. The figure contains box plots for energy consumption (in Watt hours per meter) when driving on road segments going into, inside and away from roundabouts. It shows that electric cars on average regenerate energy when driving into a roundabout, while they use noticeably more energy when driving out of a roundabout than when driving inside it or on roads not connected to a roundabout. As was done with traffic lights, the analysis is limited to the road category on which roundabouts are most common, which is also on tertiary roads, to better allow comparison between driving on road segments that do not have roundabouts at either end and on segments that do.

As was the case with the traffic lights, the presence of a roundabout is not originally indicated in our source data. To include information about roundabouts, we can use an ID attribute, `segmentid` that links a segment in our version of the OSM data to a record in the OSM data from Geofabrik[13]. In addition to points, such as those that are marked with traffic lights, this OSM data dump also contains records, in the form of lines that represent the roads themselves.

These lines each have a unique ID, `osm_id`, which we can match up with the `segmentid` column in our source data. Each of the lines in the OSM data has a `junction` attribute, which can take a value of "roundabout", indicating that the line is part of a roundabout.

Consequently, we can match segments in our source data with lines in the OSM data, and determine from this attribute whether a segment is part of, meaning inside of, a roundabout. However, as was the case with traffic lights, we are also interested in whether a vehicle is leaving or entering a roundabout. Therefore, we must annotate the segments with a value indicating whether their start- or endpoints are part of a roundabout or not. This is done with the same

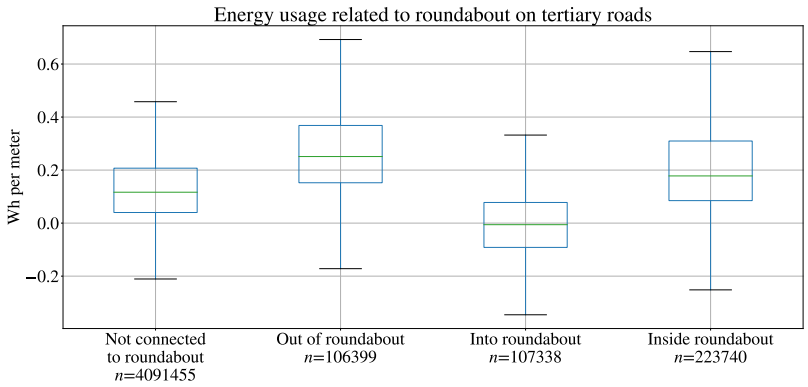


Figure 3.16: Energy consumption when driving up to, inside, and away from roundabouts on tertiary roads.

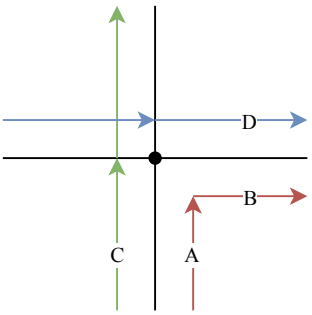


Figure 3.17: Supersegments (colored) in an intersection (black).

method as for traffic lights. The SQL queries we use for determining which points should be annotated as being part of a roundabout can be seen in Appendix B.

3.4 Angle Data

We have calculated angles between road segments that are connected in an intersection, as this may be useful context for predicting the energy consumption for vehicles that will perform turns as they move through an intersection.

Take the example of an intersection shown in Figure 3.17. The four road segments that make up the intersection are marked in black, and three trips over that intersection are marked in red, green and blue. In this example, we show parts of three trips that each include two trip segments. To reiterate, trip segments correspond to the road segments that are traversed during a given trip. For the red trip, the driver must slow down during segment A, perform a turn, then accelerate during segment B. For the blue and green trips, the driver can potentially coast before the intersection (on segment C) and after the intersection (segment D). For the green and blue trips, the driver has to turn 0 degrees when going through the intersection. If the intersection is clear, this allows the driver to continue through the intersection (segment C or

D) without decelerating or accelerating. For the red trip, the driver has to turn 90 degrees. In most circumstances, this means that the driver has decelerate before the intersection (segment A) to avoid loss of traction during the turn, then accelerate after the intersection (segment B), even if it is clear. To be able to model the effect of the turning angle on energy consumption, we must know the complete trip a vehicle will take in advance.

The relation between vehicle energy consumption and the angle of intersections between road segments can be seen from Figure 3.18. It shows the average energy consumption when driving from one road segment to the next, based on the angle of the turn from one road segment to the next. The energy consumption is calculated in groups of 1° increments. It can be seen that there is a trend of higher energy consumption when performing turns at sharper angles, up to around 40 degrees, where the energy consumption levels out. This indicates that turning above 20 degrees does not result in different in energy consumption levels than turning at 60 degrees or more. A natural explanation for this is that once it is necessary to slow down for a turn (at 20 degrees or above), turning at a sharper angle (such as 80 degrees) is almost the same as turning at 30 degrees. Angles above 130 degrees are excluded, since there are few and they mostly represent roads that are split into two roads, between which U-turns are possible.

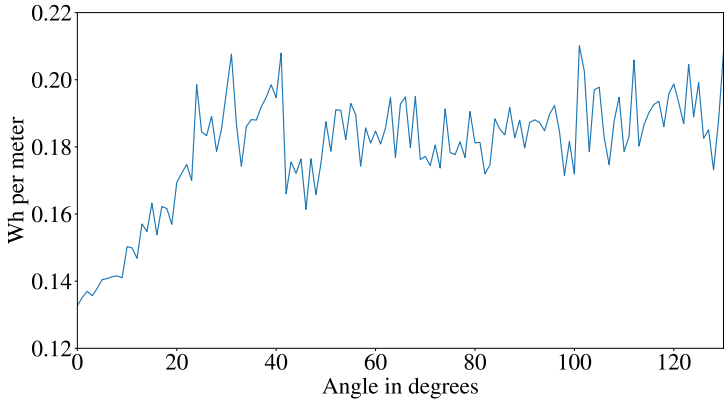


Figure 3.18: Average energy consumption after turning at a specific angle.

Our approach for calculating the angles at which vehicles turn can be seen in Figure 3.19. The figure shows geometries of road segments from OSM (red lines), points we extract from these geometries (blue dots), and the lines we construct between the points to calculate the angles (black lines). The background tile used for figures illustrating OSM geometries is the OSM tile of Denmark from 2019 [30]. For each pair of road segments that meet at an intersection, we extract the meeting point between the two segments, named B, and the closest point to the intersection for each of the two road segments, named A and C. We then extended a semi-line from A through B, then measured the shortest angle between B, C and a point after B on this semi-line (AB). Figure 3.19 shows the calculated angles for the two intersections ABC and BCD. Notice that this approach gives the same angle for ABC and CBA, and that the angles are always between 0 and 180.

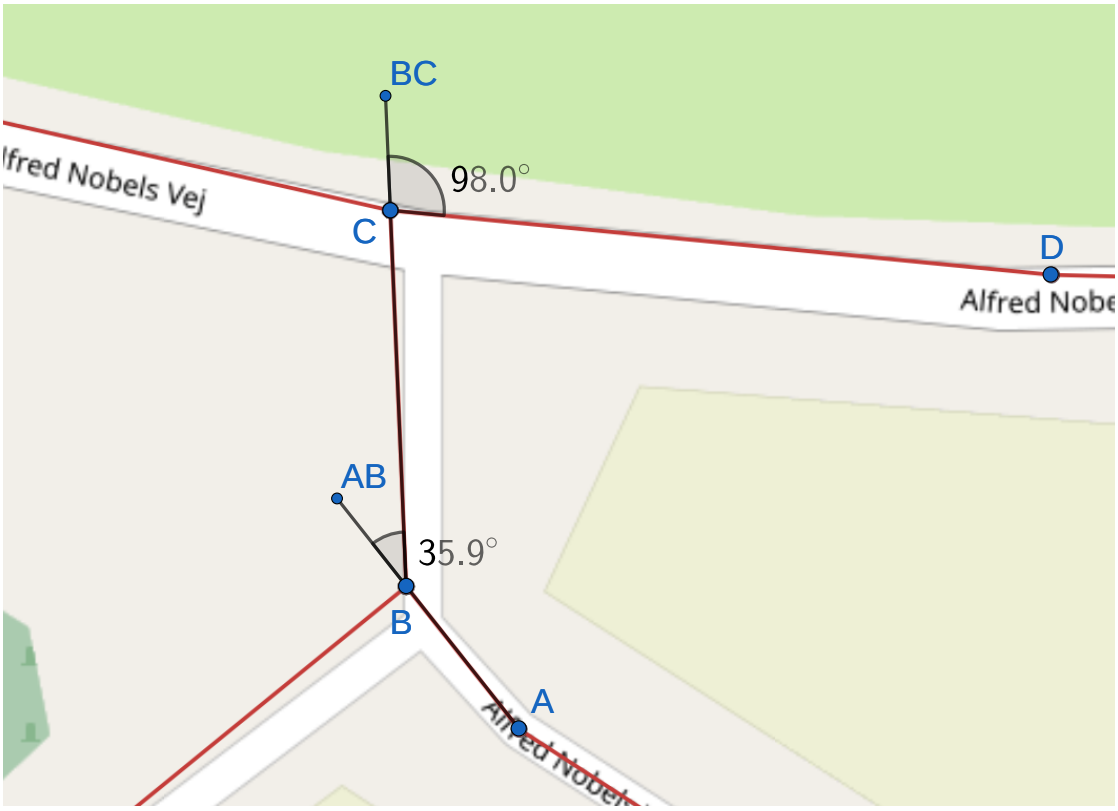


Figure 3.19: Calculation of angles of turns performed on road segments.

3.5 Speed Map Data

We have received a speed map from Vejdirektoratet (The Danish Road Directorate). The speed map consists of an ESRI shape file with road segment geometries and attached values. It only contains data for larger roads, not small residential ones. This is closely related to the category of the road, with the speed map primarily including values for categories such as primary roads or trunk roads that are the major road arteries between cities.

Each road segment has a so-called `linkid`, which is an internal Vejdirektoratet road ID and cannot be directly matched against OSM. It also has a `linkalong` Boolean value, which indicates in which direction on the indicated road the data has been collected for, relative to the direction of the line segments in the underlying geometry. Following, there can be two segments with the same geometry and `linkid` but with different `linkalong` values for a bidirectional road segment.

Additionally, each road segment has five `time` values, `time_1` through `time_5`. Each value indicates the measured speed in km/h at the indicated segment in the indicated direction at a specific time of day. See Table 3.1 for three segments annotated with average speeds at five time-intervals.

Segmentkey	00:00-06:30	06:30-09:00	09:00-15:30	15:30-18:00	18:00-24:00
164444	32	33	32.5	33.5	41
322256	49	51	44	28	48
59315	76	78	75	81	98

Table 3.1: Example values for road segments in the speedmap.

This data can be useful in predicting energy consumption, since there is a strong correlation between speed and energy consumption[7]. The speedmap data can be used to predict the speed electric vehicles travel at, illustrated in Figure 3.20. The figure shows the connection between the speed from Vejdirektoratets speedmap for trips on specific road segments at specific times of day, and the actual measures speed of those trip segments. Here, it can be seen that while electric vehicles in our dataset in general drive a little faster when the speedmap predicts low speeds, and slower when the speedmap predicts motorway speeds, there is still a linear correlation ($R^2 = 0.64$) between measured and predicted speeds in the middle of the spectrum. Please note that the linear regression is done on the raw pairs of measured speed and speed predicted by the speedmap, the graph shows average measured speeds grouped by their predicted speed in 1 km/h intervals. Additionally, note that speeds which were predicted for less than 50 trip segments (speeds above 121 km/h) have been excluded.

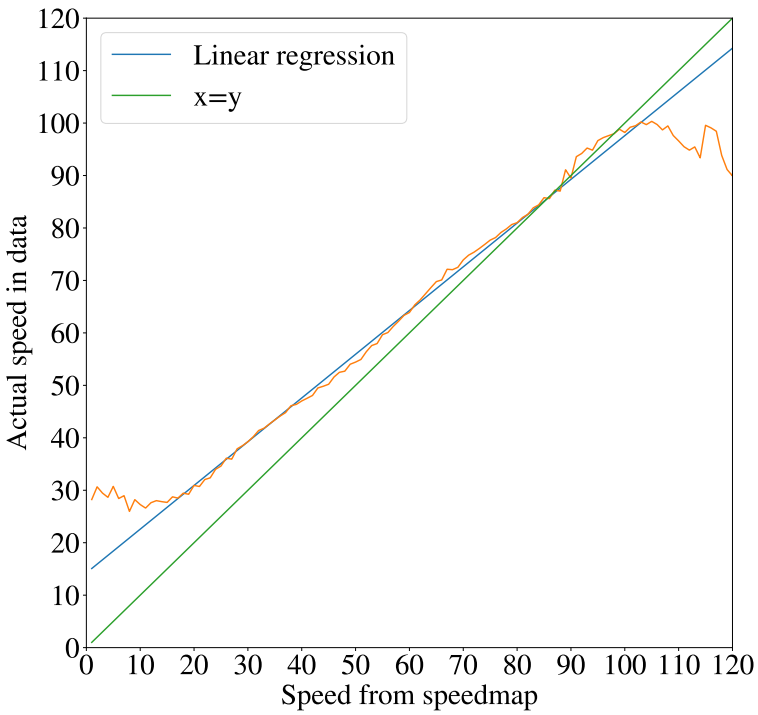


Figure 3.20: Predicted speed from the speedmap against actual speed.

Likewise, the correlation between speed predicted by the speedmap and energy consumption can be seen in Figure 3.21. This analysis shows the connection between the speed from Vejdirektoratets speedmap for trips on specific road segments at specific times of day and the measured energy consumption for those trip segments. An interesting note to take away from this analysis, is that it shows similar trends to the one for the actual speeds from our speed analysis (Figure 3.5). In Figure 3.21 it can be seen that energy consumption per meter is higher at low speeds, again possibly due to the higher energy consumption caused by accelerating or due to the use of auxiliaries. However, from 40 km/h to around 105 km/h, energy consumption per meter grows almost linearly with predicted speed. The trend conforms with what we saw from the actual speed analysis, however the speed predictions alone do not predict energy consumption. Even though this is not the case, speed predictions can nonetheless be an important contributor to energy predictions[7]. Note that, like for Figure 3.20, speeds which were predicted for less than 50 trip segments (speeds above 121 km/h) have been excluded.

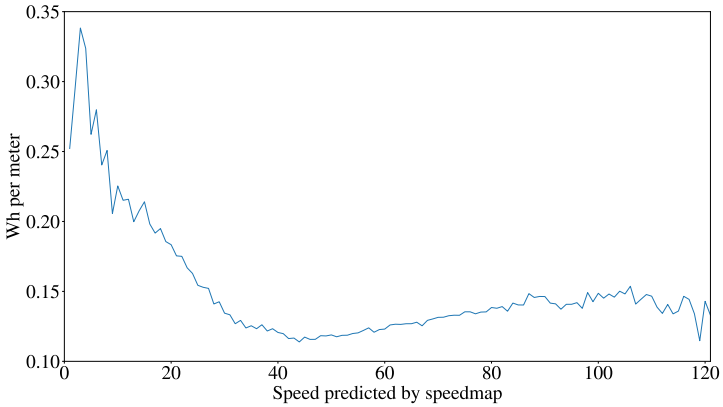


Figure 3.21: Correlation between speedmap speed and energy consumption.

For us to be able to use the speed map from Vejdirektoratet in our model, we have to assign the measured speeds to the OSM map that is used for our trip data. These two maps are not modeled the same, meaning that the same stretch of physical road may be modeled by more or fewer lines in the speed map, compared to the OSM map. Smaller roads in OSM will not have any counterpart in the speed map. Additionally, there is an often changing offset between the lines in the two maps. The difference is shown visually in Figure 3.22. Note that the red OSM segments are not aligned with the background map, as the map is a newer version. The arrowheads show the ends of each line segment, and it can be seen that it varies between multiple segments from the speed map covering the same stretch of road as one OSM segment, and vice versa. Due to these challenges, it is necessary to implement some kind of matching between the segments in OSM and the segments in the speed map.

We handle this challenge by adding a 10 meter radius buffer around the line segments for both the speed map and OSM. For each OSM segment, we then match up all speed map segments for which the area of the intersection between the buffers is at least $10^2 \cdot \pi \cdot 1.5$. This ensures that segments which only meet at an intersection (and should therefore have a buffer overlap of approx. $10^2 \cdot \pi$) are not matched up. Last, we assign speeds for each of the five time periods to each OSM segment as an average of the values for matched speedmap segments weighed by the area of the buffer overlap.

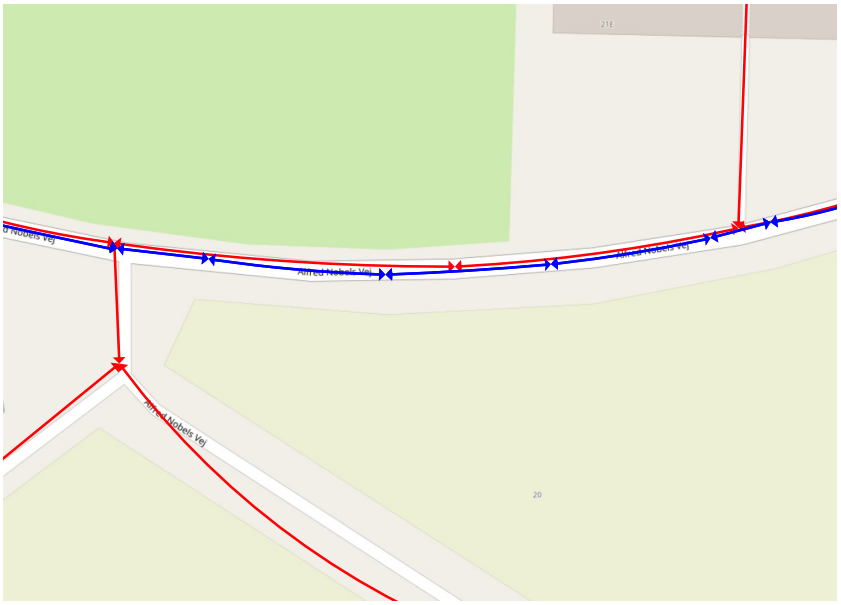


Figure 3.22: Segment differences between speed map (blue) and OSM (red).

An example of this method can be seen in Figure 3.23. It shows the buffers used to match the segments and calculate average speeds for OSM segments. The OSM buffers are shown in red, while the speedmap buffers are shown in blue. Note the two OSM segments A and B (diagonally stribed), which meet a speedmap segment at an intersection, but which should not have speeds assigned. Also note the OSM segment C (horizontally stribed), which overlaps two speedmap segments, and which is therefore assigned a weighted average of the speeds of those two segments.

3.6 Energy Consumption Label

For this project, we wish to predict the energy consumption of electric vehicles, based on the context of the vehicle at the time of driving on a road segment. We build upon the work by Bundgaard et al. [7] by extending the dataset with information about the road network, in the form of average speeds from the speedmap provided by Vejdirektoratet, turning angles and the presence of traffic lights and roundabouts. These new additions are related to the road network itself, and we have shown that each feature individually is correlated with changes in the energy consumption (Figures 3.14, 3.16, and 3.18), or are strong predictors of the speed that a vehicle will drive at (Figure 3.20).

We include this information to be able to more precisely model the energy consumption of the vehicles in our dataset, specifically by using an energy consumption attribute present for trips as a label for supervised learning. In Andersen et al. [1, 2] it is explained that the energy consumption values for our dataset, for trip segments, are based on power consumption readings from GPS points that are map-matched to a specific road segment. This means that the energy consumed over a specific road segment for a specific trip is calculated from the GPS points that are matched to that road segment for that trip.

During an analysis of possible errors in the energy consumption readings, inspired by the work of Bundgaard et al. [7] which highlights that 0.5% of all trips have a negative energy consumption,

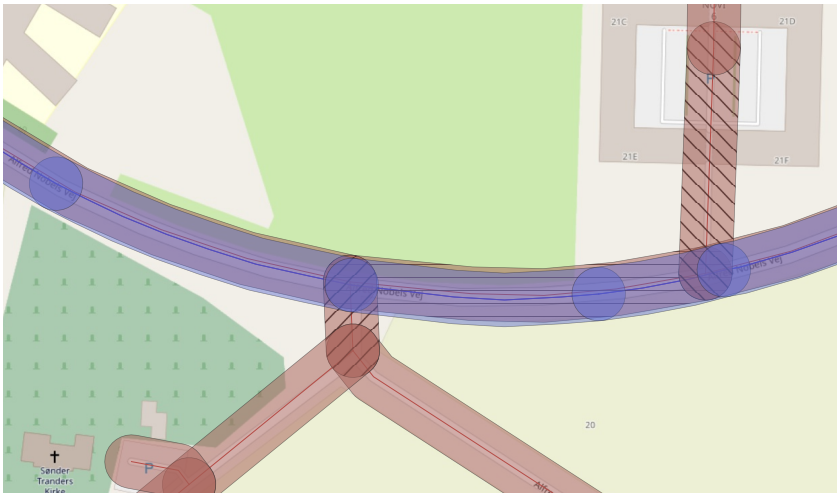


Figure 3.23: Buffers around OSM and speedmap segments.

we have found that there are additional trip segments with highly unrealistic energy consumption figures. In one instance, a vehicle traversed a segment with a length of 21 meters over 170 seconds, expending 1.6 kWh or approximately 10% of the battery capacity of the vehicle. This requires an average output of 35 kW for the whole duration, which is highly unrealistic. We found this to be caused by the calculation of the `ev_kwh` attribute defined in Figure 3.1, from the GPS points previously map-matched to each road segment by Krogh et al. [22].

We therefore recalculate the `ev_kwh` attribute, and for this we use the `ev_watt` attribute on the GPS points that are map-matched to each trip segment. We follow the rules set in Andersen et al. [2] for `ev_watt` (named `EVPower` in the report), disregarding any measurements above 50 kW and below -20 kW. These figures are set as a maximum and a minimum for the energy consumption and regeneration respectively, that these vehicles are realistically capable of. The query for recalculating the `ev_kwh` attribute can be seen in Appendix C. For this purpose, we assume that the value provided by the `ev_watt` attribute is the energy consumed over the previous second, in the form of Watt seconds, as the GPS data is recorded at a frequency of 1 Hz. This is converted into kWh by the formula $= 1$ kWh $= 60 \cdot 60 \cdot 1000\ W/s$. We first sum all valid `ev_watt` values for each trip segment, and then convert from Watt seconds to Kilowatt hours by dividing the sum by $60 * 60 * 1000$, as there are 3600 seconds in an hour, and 1000 Watts to a Kilowatt.

With the recalculated `ev_kwh` for trip segments, 0.026 kWh was expended for the previously mentioned trip segment. A manual inspection indicates this to be a much more likely figure, as 120 of the 170 seconds is spent idling at the same spot with an average output of 550 Watts for the whole duration.

Furthermore, with the upper- and lower bounds, we have filtered away the impact of incorrect readings for all trip segments, obtaining more realistic energy consumption figures. The new label is likely to improve our model as well, as the new `ev_kwh` figures will be more representative of the actual energy consumption for a trip segment. All previous data analysis for determining correlation with energy consumption of electric vehicles has been performed using this new label.

Chapter 4

Models

As mentioned in Chapter 2, Zhang et al. [43] structure their data in such a way as to exploit the contextual nature of trip sequences, for travel time estimation. This allows them to capture interactions between movement from one grid cell to another that affect the travel time of the trip. Inspired by this approach, we will consider trips in our dataset as a sequence of trip segments, and create models that can capture cross-segment interactions within each trip.

The model developed by Bundgaard et al. [7], which we use as the baseline for this report, is implemented as a Deep Neural Network (DNN). This model does not consider any context, predicting the energy consumption of a trip segment with no regard for the impact of surrounding trip segments.

One possibility for capturing this context is to, for each trip segment, concatenate the features from surrounding trip segments. The energy prediction is still performed for individual trip segments, however, it now considers interactions between the features of the surrounding trip segments and the energy expended over a trip segment. We call this method of including contextual information for a DNN "supersegments".

Another possibility for training on sequential data to include context is a recurrent neural network (RNN), which is a variant of a neural network. An RNN takes a three-dimensional input, with the additional dimension being time, and trains on full sequences or parts of sequences with a sliding window consisting of a number of time steps. A standard RNN implementation will only propagate information from previous time steps, however it is possible to implement a bidirectional RNN which propagates information from both directions.

In addition to the improvements on the Bundgaard et al. [7] model to include context, we can also include more sophisticated handling of some data sources. This can include incorporating a pretrained speed prediction model into the energy prediction model, hereby using it to learn an underlying representation of the patterns related to vehicle speed, but then allowing the energy prediction model to change to weights to optimize for energy prediction. Additionally, it can include embedding categorical features into dense feature vectors instead of including them as one-hot encodings, thereby learning similarities between different categorical values.

4.1 Baseline Model

The baseline model for this project uses the feature set shown in Appendix D. It is the model created by Bundgaard et al. [7]. The data used by this model does not include contextual information such as angles to surrounding segments, or features from these. Of special note are the categorical features, which are not input directly to the model, but are instead transformed

prior to their use by one-hot encoding them. One-hot encoding a feature means creating a vector with a number of indices equal to the number of categories. Membership of a category is then denoted with a 1 in the category's index in this vector, and 0 for all other categories. We also add a spatial embedding of the road network, created with node2vec, to this feature set. Finally, we also include a predicted speed, obtained by first training a model on a speed label, using a similar model to the baseline energy prediction model.

As a preprocessing step, we standardize the continuous features, if they are not already standardized, such as the node2vec embeddings. This is to reduce the impact of outliers within a feature, such as for the length of a segment. This is to reduce the difference in weights for features with different scales. As an example, the length of segments vary from less than a meter to several thousand meters, while incline is at most 10 percent positive or negative. Each feature value is scaled as

$$z = \frac{x - \tilde{x}}{\sigma},$$

where $\tilde{x}$ is the feature mean, and σ is the variance.

4.1.1 Preprocessing

The features for the model come from multiple different datasets, including one with trip segments, one with speed predictions, and one with the road network embeddings. Conducting experiments with different feature subsets, number of context segments, embeddings, etc. requires preprocessing and joining these datasets many times in different ways. Due to the size of the combined dataset, we need the preprocessing pipeline to be fast and efficient.

The memory usage needs to stay within the limits of the primary memory installed in the computers we run the experiments on to avoid swapping. The step that increases memory usage the most is joining the graph embeddings onto the trip segments, as the embedding for each road segment is duplicated across the many trip segments on that road segment.

Following, our goal for the pipeline is that unused data is dropped as early as possible, data that is used separately is extracted as early as possible, and the merge with the spatial embeddings is performed as late as possible. This leads us to a pipeline, seen in Appendix E, with the following characteristics:

1. Loading only data that is going to be used into memory and loading it as late as possible
2. Extracting labels, join keys, and data that is to be transformed to avoid duplicating them in memory during processing
3. Dropping join keys and other data as soon as it is no longer useful

This pipeline architecture means that datasets for small context windows can be fully contained in memory and that only few pagefile interactions are necessary for large context windows. Therefore, the full preprocessing can run in a few minutes.

4.1.2 Spatial Graph Embeddings

This section is adapted from [7]. Bundgaard et al. [7] found that graph embeddings generated with node2vec improve the quality of energy predictions. For this reason, we use node2vec to generate spatial embeddings for segments in the road network. For the purpose of running node2vec, we model the road network as an undirected graph $G = (V, E)$. In this graph the edges are the road segments and the vertices are the intersections between the road segments.

However, node2vec is node-centric, which means that it embeds the nodes of a graph. As we wish to embed the road segments, which are the edges of our graph, we must first obtain the *dual graph* of our road network graph. The dual of a graph is the graph in which the edges from the primal graph have been converted to nodes. The pseudocode for inverting the graph in this way can be seen in Algorithm 2. The e.asNode () method returns the edge e as a node while retaining all its features.

Algorithm 2 Calculates the dual graph of a graph G.

Input: Graph of road network $G = (V, E)$.
Output: Dual graph of road network $G' = (V', E')$.

```
function TRANSFORM(G)
    V' ← {}
    E' ← {}
    for all e ∈ E do
        V' ← V' ∪ {e.asNode ()}
    end for
    for all e1, e2 ∈ E do
        if e1 ≠ e2 and (e2.asNode () , e1.asNode ()) ∉ E' and e1 shares a node with e2 then
            E' ← E' ∪ {(e1.asNode () , e2.asNode ())}
        end if
    end for
    return G' = (V', E')
end function
```

After generating the dual graph, it is possible to generate embeddings for the road segments, which are now nodes of the graph, with node2vec.

node2vec

node2vec is a node embedding method proposed by Grover and Leskovec [14]. It extends Skip-Gram for use on graphs. Let $G = (V, E)$ be a network graph. The goal of the model is to learn a mapping function $f : V \to \mathbb{R}^d$ from nodes in a graph to feature representations. Based on a notion of the neighborhoods $N_S(u) \subset V$ of each node $u \in V$ sampled with a sampling strategy S, the objective when training the model is to optimize the equation:

$$\max_f \sum_{u \in V} \left[-\log Z_u + \sum_{n_i \in N_S(u)} f(n_i) \cdot f(u) \right],$$

where $Z_u = \sum_{v \in V} \exp(f(u) \cdot f(v))$ is the per-node partition function (the likelihood that each node $v \in N_S(u)$), which is expensive to compute for large networks, and is therefore approximated using negative sampling. Negative sampling means sampling from the set of nodes not in the neighborhood of u, rather than computing a softmax over the entire set of nodes. Intuitively, the $\sum_{n_i \in N_S(u)} f(n_i) \cdot f(u)$ part of the equation incentivizes assigning similar feature values to nodes in each others neighborhoods, while the $-\log Z_u$ part of the equation incentivizes assigning dissimilar (orthogonal) feature values to nodes that are not in each others neighborhoods.

The sampling strategy used in node2vec samples the neighborhood for a source node $u = c_0$ by simulating a second order random walk of a fixed length, $c_0, \ldots, c_l$, generated by the probability distribution:

$$P\left(c_i = x \mid c_{i-1} = v\right) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v,x) \in E \\ 0 & \text{otherwise} \end{cases}$$

where π_{vx} is the un-normalized transition probability between nodes v and x, and Z is the normalizing constant. The transition probability is defined based on a search bias α as $\pi_{vx} = \alpha_{pq}\left(t,x\right) \cdot w_{vx}$, where t is the node from which the previous transition to v was taken, w_{vx} is the edge weight (or 1 if the graph is un-weighted), and the search bias α is defined as:

$$\alpha_{pq}\left(t,x\right) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

where d_{tx} is the shortest-path distance between nodes t and x.

The search bias is parameterized with two parameters p and q. The parameter p, called the return parameter, determines the likelihood of the random walks revisiting the previous node, with higher values encouraging exploration of nodes other than the one that was just visited. The parameter q, called the in-out parameter, biases the walks toward nodes closer or further away, with higher values encouraging transitions to nodes that are direct neighbors of the previous node on the walk. The authors of node2vec describe how varying the value of the q parameter allows the random walk to behave more like a Depth First Search (DFS), for low values, or a Breadth First Search (BFS), for high values. This allows the algorithm to produce embeddings that more closely indicate communities based on homophily, for DFS, or structural equivalence, for BFS [14].

The node2vec [14] implementation we use is publicly available from the Stanford Network Analysis Project (SNAP), and is created in C++ [23]. node2vec is run on the dual graph of our road network, to obtain a node embedding for each road segment. We set the size of the embeddings to 64, the walk length to 20 and we keep the number of negative samples at the standard value of 5. This is in accordance to the parameters used by Bundgaard et al. [7]. The embedding vectors created by node2vec are used in our model by concatenating the vector to the set of input features for each trip segment.

4.1.3 Architecture

We construct the baseline as a DNN, as these are capable of learning complex and nonlinear interactions between the input features and the label. A DNN is constructed by stacking a varying number of fully connected layers between the input and the output layer. These layers are called hidden layers, and serve as intermediate representations following nonlinear transformations of the input. The output of the network is a vector. In our case, the output is a scalar, and the goal is to train the network to model the energy consumption over a segment, using the input specified in Appendix D.

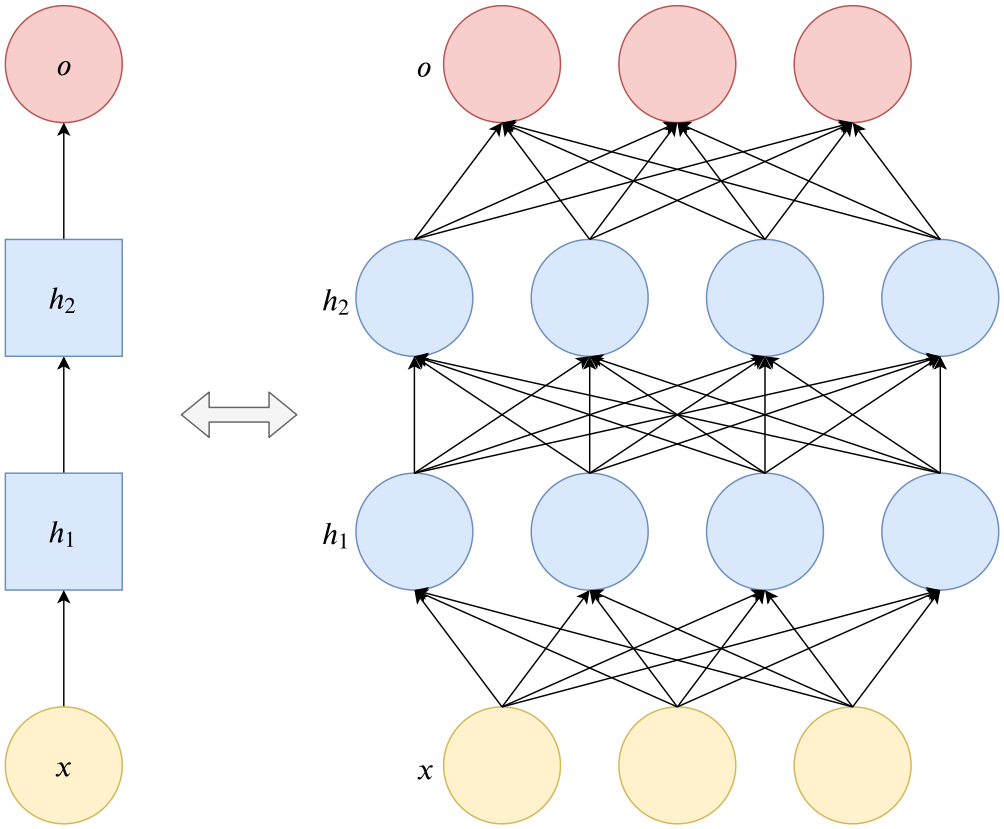


Figure 4.1: A DNN with two hidden layers, shown in matrix format (left) and scalar format (right).

The calculations of a neural network can be described by matrix operations of the form

$$h_i = \sigma\left(W_i^\top x_i + b_i\right).$$

Here, $h_i \in \mathbb{R}^N$ is the output of layer i in the network. $x_i \in \mathbb{R}^M$ is the input for the layer with $x_1 = x$ being the input to the network and $x_i = h_{i-1}$ for $i > 1$. σ is the activation function, $W_i \in \mathbb{R}^{M \times N}$ is the weight matrix, and $b_i \in R^N$ is a bias term for the layer h_i. The bias term can optionally be eliminated by adding a constant 1 entry to the end of the input vectors and adding an extra row and column to the weight matrix with the last entry being the bias and all other entries being 0.

In Figure 4.1, an example of a DNN can be seen with two hidden layers (h_1 and h_2), an input layer (x), and an output layer (o). The example includes two common ways of visualizing DNNs. The left side is the matrix form, where the input layer and each matrix multiplication step described is represented as one object. The right side is the scalar form, which explicitly shows the dimensionality of the output of each layer by having one object per entry in the output vectors. Thus, each object represents multiplying each input scalar (the arrows) by their

corresponding weight, summing these products, and mapping the output scalar through the activation function.

For the example the output of h_1 would be

$$h_1 = \sigma\left(W_1^\top x + b_1\right),$$

with $W_1 \in \mathbb{R}^{3\times4}$. This series of operations is then repeated for h_2, with the output of h_1 as the input, weight matrix $W_2 \in \mathbb{R}^{4\times4}$, and the bias term for the layer. Lastly, it is repeated for the output layer o with input h_2 and weights W_o.

The choice of activation function for the output layer o depends on the task. For our regression task, we use the identity function, meaning that the output is simply the sum of the inputs. For other tasks like binary classification, other activation functions, such as the sigmoid activation function, can be used.

A DNN defines a number of hyperparameters that can be adjusted for the purpose of improving the performance of the network. These are the number of layers, the size of each layer, activation functions used to introduce nonlinearity, and the method of training the network. For our experiments, we vary the network size hyperparameters of the baseline model in the following ranges: We use between two to seven hidden layers, and we start from 250 units, doubling the number of units every step until 2000. The use of activation function (σ) is selected to be the Rectified Linear Unit (ReLU) and remains static throughout our experiments. ReLU introduces nonlinearity to the network by transforming the input to a neuron in the hidden layers and is a commonly used activation function for neural networks. ReLU is defined in Equation (4.1) and is visualized in Figure 4.2.

$$f(x) = \max(0, x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}. \tag{4.1}$$

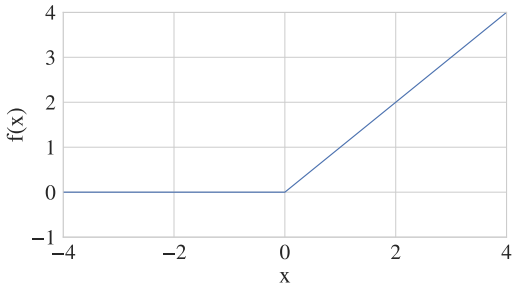


Figure 4.2: ReLU function

Loss functions

The performance of the DNN is measured by a loss function, which the goal is to minimize. This loss function is defined based on the differences between predictions made by the DNN for a sample and the actual label for that sample.

An example of a loss function commonly used for regression is Mean Squared Error (MSE). This loss function, defined in Equation (4.2), penalizes large differences by squaring the difference

between the prediction $f(x)$ and the label $f'(x)$ for each sample x in a dataset X. As a result, MSE penalizes few large mistakes more than many small mistakes.

$$MSE = \frac{1}{|X|} \sum_{x \in X} \left(f\left(x\right) - f'\left(x\right) \right)^2. \tag{4.2}$$

Other popular loss functions for regression are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). RMSE is defined by taking the square root of MSE. It therefore also penalizes large differences more, but gives a result that is in the original unit, which makes RMSE more easily interpretable than MSE. MAE is defined in Equation (4.3). Unlike the two other loss functions, MAE does not penalize large differences between the prediction and the true value and instead uses the absolute difference between the two. This means that MAE penalizes all mistakes linearly, based on their magnitude. MAE is also in the same unit as the prediction and the label values, meaning that MAE is more easily interpreted.

$$MAE = \frac{1}{|X|} \sum_{x \in X} \left| f\left(x\right) - f'\left(x\right) \right|. \tag{4.3}$$

Training

The goal of training or optimizing the model is to minimize the average loss of the model on samples. When training the model, we use the MSE loss function.

The model is optimized by gradient descent using backpropagation. Gradient descent consists of calculating a gradient, that is the change in loss for a specific training sample when changing the weights of the neural network. The weights are then gradually changed in small steps by iterating over the training dataset, calculating the gradient, and updating the weights in the direction that minimizes the loss.

Formally, the gradient is a vector consisting of, for each weight, the partial derivative of the loss function (with the current training sample and weights as input) with regards to that weight. The weights are updated by multiplying the gradient with a learning rate and subtracting it from the current weights. Backpropagation is an efficient way to compute the gradients used for gradient descent, moving backward from the output layer and propagating towards the input layer.

For DNNs, the use of gradient descent for training can result in two well known challenges, known as vanishing gradient and exploding gradient. A vanishing gradient means that a change of a weight in a DNN has a very small effect on the loss, resulting in a very small gradient. The cause of vanishing gradients in DNNs is the use of activation functions that are bounded in a certain range, such as $[0, 1]$ in the case of the sigmoid activation function. These types of activation functions become "saturated" at large positive and negative values, where their derivatives tend to 0, meaning that large changes in the input only changes the output slightly towards or away from the bounds. In the case of the sigmoid function, large negative values are squeezed towards 0 and large positive values towards 1. This means that, when the multiplication of weights and training samples lead to absolutely large activation function inputs, the calculated gradient will be vanishingly small, preventing the model from learning from these samples. Vanishing gradients occurs more frequently for the weights closer to the input of the DNN, as the effect of squeezing through saturated activation functions throughout the network compounds the issue, resulting in ever smaller gradients at the start of the network. Our choice of activation function, ReLU, is shown to reduce the likelihood of experiencing vanishing gradients for deep neural networks due to the fact that it only saturates in one direction, as it has no positive bound[41].

Exploding gradients is the opposite problem, where the gradients have a very large magnitude, meaning that changes in a weight has a large effect on the loss. This leads to large changes in weights, rather than small incremental steps, preventing convergence or leading to overflows if the weights are stored as floating-point numbers. The ReLU activation function also reduce the likelihood of this problem, as the derivative of the function is always either 0 or 1. Additionally, it is possible to clip the gradients by decreasing their magnitude, or alternatively, constraints can be set on the weights of the network by use of weight regularization. The regularization is commonly implemented as a term added to the loss function, with a parameter λ, which controls the amount of regularization.

For our models, we experiment with L2 regularization, which controls the size of the weights in the network. L2 regularization penalizes large weights w by adding the square (w^2) of the weight value to the loss function when calculating the gradients, scaled by λ. An increased amount of L2 regularization (increasing λ) then forces the model to use lower values for its weights, due to penalizing larger weights more. This both reduces the risk of exploding gradient, and can additionally help the model generalize better, preventing over-fitting, by encouraging the model to use all parts of the input in predictions. In Equation (4.4), we show the MSE loss function with a regularization term.

$$MSEReg = \frac{1}{|X|} \sum_{x \in X} \left(f\left(x\right) - f'\left(x\right)\right)^2 + \frac{\lambda}{|X|} \sum_{i=1}^{n} w_i^2, \tag{4.4}$$

where n is the number of weights and w_i is the value of weight number i.

In practice for training our neural networks, we use a variant of gradient descent called stochastic gradient descent with mini-batching. Here, small, random batches of for example 1000 training samples are grouped together, the gradients for each of them is calculated in parallel, and then a step is taken based on the average of the gradients.

An optimizer is used to determine exactly how much to change the weights based on the average gradient for the batch. The normal stochastic gradient descent optimizer takes a fixed learning rate as a hyperparameter and multiplies this learning rate on the gradient before subtracting it from the weights. Momentum optimizers keep a "memory" of the weight changes used in previous iterations, and applies some part of this change again in following iterations. This helps avoid the gradient descent getting stuck in small local minimums, allowing it to "roll" over minor bumps on the downwards slope.

We use a variant of the Adam momentum optimizer called Adamax. The Adam optimizer, or **Ada**ptive **M**omentum optimizer, changes the amount of momentum retained based on the magnitude of the updates performed during previous iterations. The Adamax variant was shown by Bundgaard et al. [7] to perform well for this type of dataset.

Normally, training is performed repeatedly on the full dataset until the weights converges to a point where the loss doesn't change. Each of these training iterations on the full dataset is called an epoch, and we train for a number of epochs until the model does not improve its performance on a validation dataset.

Evaluation

For the purpose of evaluating the model, we construct three datasets from our complete dataset. These three datasets are the training, validation and test datasets and have a ratio of approximately 5:2:3 leading to a 70/30 split for training/validation and testing data. We use 49% of the data for training, 21% of the data for validation of the model, and the remaining 30% is kept separate for final model evaluation. When evaluating the final model after the parameter search, we train on the combined training and validation sets, meaning 70% of the data, and evaluate

on the remaining 30% test data. This split is the same split used by Bundgaard et al. [7], and is obtained by randomly sampling trips in our full data and placing them in the training, validation or the test dataset in the aforementioned ratios.

4.2 Supersegments

Supersegments is our first approach to adding contextual information about surrounding segments to the dataset, and uses an architecture identical to that of the baseline model. The context is added to the data used by the model, allowing an arbitrary number of trip segments before or after the focus segment to be used as context, while still using the same model architecture. For example, with a context windows of three segments, the segment just before and after the focus segment are included. The window can then be moved over a trip incrementally, to predict the energy consumption for each individual segment by including data from the neighbors within the window.

4.2.1 Data

The supersegment model uses the same features as the baseline model for each segment, including the embeddings obtained by node2vec and a speed prediction. In addition, it includes the angle between each pair of segments as a new contextual feature, as described in Section 3.4. The names of the features of each trip segment that is part of the context window are prefixed, denoting which trip segment they belong to. For example, for a context window of three segments, the `incline` feature of the three segments would be named `s01_incline`, `s02_incline`, and `s03_incline`. The features of the segments are then concatenated together into one feature vector with the features of the focus segment in the center. When training the model, each of those segments would be the center segment in one training iteration. This is shown for a simple dataset with two features in Table 4.1, where each row represents a feature vector.

s01_incline	s01_temp	s02_incline	s02_temp	s03_incline	s03_temp
0.000	0.000	1.234	21.23	2.345	22.34
1.234	21.23	2.345	22.34	3.456	23.45
2.345	22.34	3.456	23.45	0.000	0.000

Table 4.1: Example of formatted training data for a context window of 3 segments.

For the first and last trip segments of a trip, where there are no previous or next segments to include, zeroed out features for these segments are used instead, as shown in the first and last row. The model still only aims to predict the energy consumption for the focus segment, and as such, only the label for the focus segment is kept.

4.2.2 Architecture

Like for the baseline model, the architecture consists of an input layer, a number of fully-connected hidden layers, and an output layer. Only the input layer differs here, as the input grows linearly with the amount of context segments that is included. The model is also trained in the same way, with the MSE loss function based on the label for the focus segment. The hyperparameters that we tune for this model is the amount of hidden layers and units, as well as the amount of L2 regularization. We experiment with a size of two to seven hidden layers, and

250 to 2000 units per layer, as with the baseline model. For the L2 regularization we experiment with λ values of 0.0, 0.005, 0.01 and 0.02. For the supersegment model we only experiment with context windows with the same number of trip segments before and after the focus segment. We experiment with context window sizes of three and five.

4.3 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a type of neural network which can learn patterns in sequential data by modelling it as a series of *time steps*. A time step in our case corresponds to the input describing a trip segment with subsequent processing of that input. The RNN learns dependencies by propagating information between the hidden state at each time step. This means that RNNs can be modelled as a neural network with a loop that propagates information from the hidden state of previous time steps to future time steps. In Figure 4.3, we show an example of an RNN, as well as version that is unrolled to three time steps.

The calculations of a recurrent neural network can, similarly to a non-recurrent network, be described by matrix operations of the form

$$h_i^t = \sigma\left(W_i^\top x_i^t + U_i^\top h_i^{t-1} + b_i\right).$$

Here, $h_i^t \in \mathbb{R}^N$ is the output in time step t of layer i in the network, or a zero vector for $t = 0$. $x_i^t \in \mathbb{R}^M$ is the input in time step t for the layer with $x_1^t = x^t$ being the input to the network in time step t and $x_i^t = h_{i-1}^t$ for $i > 1$. σ is the activation function, $W_i \in \mathbb{R}^{M\times N}$ is the weight matrix for the input, $U_i \in \mathbb{R}^{N\times N}$ is the weight matrix for the output of the previous time step, and $b_i \in R^N$ is a bias term for the layer h_i. For RNNs, the weight matrices W and U, as well as the biases b are shared between all time steps.

The idea is that reusing the output from the previous time step ($U_i^\top h_i^{t-1}$) can allow the network to learn information that stems from the order of the sequence. As an example, information from the hidden state h_1^2 with its corresponding inputs further informs the output for time step 3 (o^3) by being included as input to the hidden state h_1^3 in Figure 4.3.

Multiple types of RNN units exist which calculate the hidden state passed on to the next time step differently from the output passed on upwards in the network. Among these, the widely used Long Short-Term Memory[16, 11] (LSTM) and Gated Recurrent Unit[8] (GRU) cells specialize in learning long-term dependencies in the data.

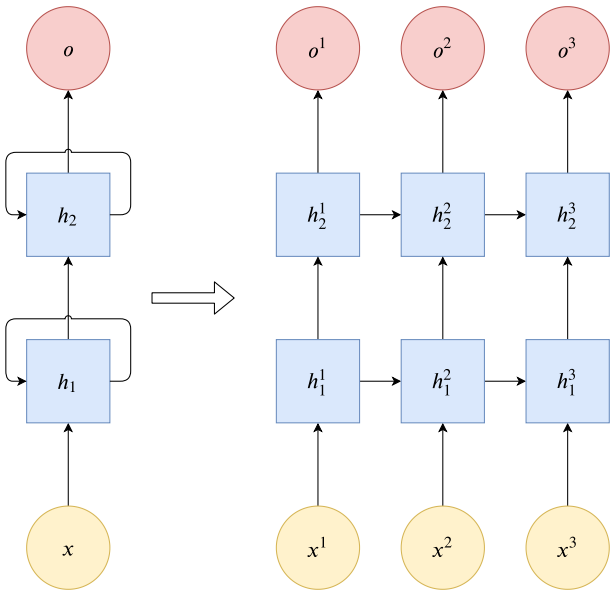


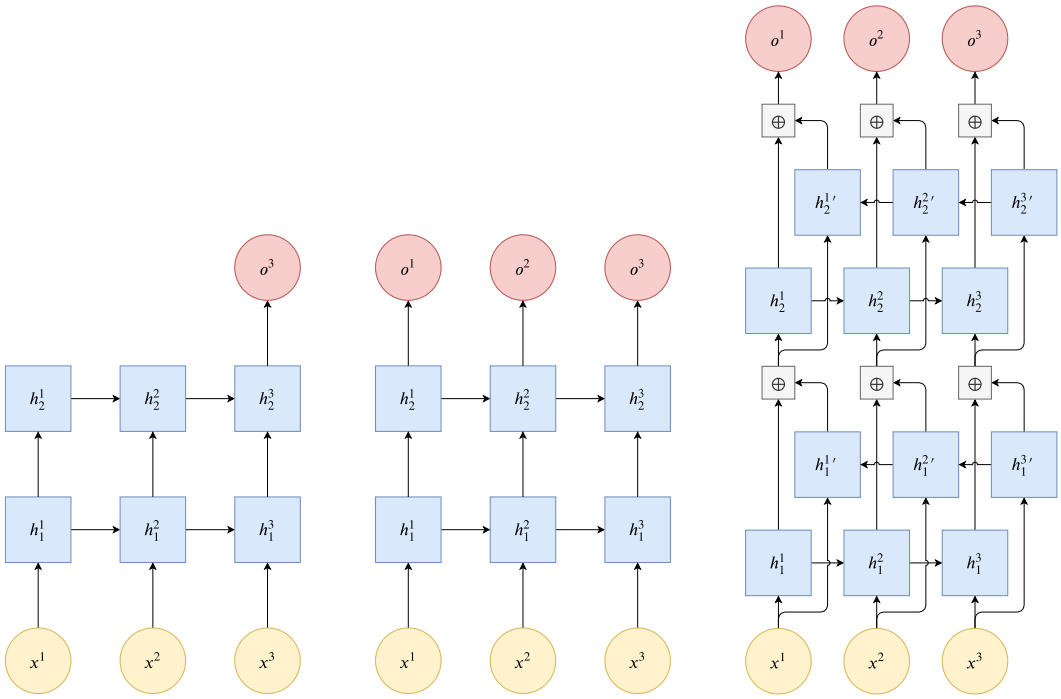
Figure 4.3: An RNN and an unrolled representation with three time steps.

RNNs are trained via gradient descent just as DNNs. However, calculating the gradient requires taking into account that the output of a unit depends both on the input from the previous layer and the hidden state of the previous time step, both of which may recursively depend on output of previous layers and timesteps. This is handled with a variant of backpropagation called backpropagation through time.

As highlighted in Section 2.3, we can model trips in our dataset as a sequence of trip segments. This means that, for our data, a time step represents the data for one trip segment. At each time step, the RNN receives the data describing a trip segment as input, as well as the information propagated from the previous time step. RNNs can propagate the information from sequences in various ways, shown in Figure 4.4. Figure 4.4a shows a many-to-one approach, in which the RNN learns information from the input sequence, to make a prediction for the final time step. For our dataset, this is equivalent to looking at the two *previous* trip segments as context for predicting for the focus segment. An alternative to this approach is to reverse the input sequence, to look at the two *next* trip segments.

Figure 4.4b shows a many-to-many RNN which outputs a prediction for every time step. This allows the model to learn from more predictions in a single sequence, with a varying number of time steps as context. For the first time step, there is no previous information to be propagated. Following that, the hidden information from $\mathtt{H}_1$ is propagated to inform the prediction for the second time step, which is then further propagated to the third time step. Figure 4.4c shows a bidirectional variant of a many-to-many RNN. Here, the hidden layer is split into a forward part, and a backwards part. The input for each time step is given to both the forward and backwards parts, whose outputs are then concatenated. Allowing information to flow from the start of the sequence to the end, and vice versa, allows each of the time steps to have equal amounts of context. It also allows the network to learn interactions from one end of the sequence to the other, that do not appear the other way around. For trip segments this could, for example, be

that the features of "future" trip segments are more relevant for determining energy consumption, or alternatively the other way around.



(a) A many-to-one RNN.

(b) A many-to-many RNN.

(c) A bidirectional many-to-many RNN.

Figure 4.4: Three different RNN architectures.

4.3.1 Data format

The RNN model uses the same data as the supersegments approach, and applies mostly the same processing, with a couple exceptions. For an RNN, we have to model the input data as a sequence rather than a larger number of input features to the model.

The data is split by the prefixes shown in Table 4.1, and are then used to create a 3-dimensional dataset where the third dimension is the time steps. For a many-to-many RNN, this also means that we need to have a label for each time step, as each output is used for training the model, rather than just a label for the focus segment.

We argue that the context we wish to extract from the sequences is local, as the energy consumption we wish to predict for a specific trip segment has no dependencies on the energy consumption of segments that are far back in time or in the future. To illustrate this logic, imagine a trip of a few kilometers in length which includes travel within a city such as Viborg, followed by driving on motorways to Aalborg. Leaving Viborg to enter the motorway to Aalborg is characterized by first reaching a motorway link, prior to driving on the motorway, on which acceleration from city or countryside speeds to motorway speeds happen. On the motorway, the context of having driven within Viborg towards the motorway link is not relevant for predicting

the energy consumption. The important context when driving on the motorway is whether the previous segment was also a motorway segment, or a motorway link.

A similar logic can be applied for future segments. If the planned route includes driving within Aalborg, the most relevant part for predicting energy consumption on the motorway is whether the following trip segment continues on the motorway or takes an off-ramp, i.e. a motorway link. For this reason, we believe the simplest method for including relevant context to be an implementation that uses a rolling window over the trip sequences. With this windowed approach, we utilize the most relevant part of each trip to inform the energy consumption prediction. This also significantly simplifies implementation complexity, since all sequences will be of identical length. This is in contrast to a model which learns on entire trips, which can vary significantly in length. For our dataset, the difference between the number of segments in the longest actual trip and the average trip is approximately 650 segments.

Due to this focus on local dependencies in the data, we choose to implement an ordinary RNN instead of one using LSTM or GRU cells.

4.4 Additional Architecture Implementations

4.4.1 Pretrained Incorporated Speed Model

As decribed in Section 4.1, we generate speed predictions by training a model with speed as the label, and then use those speed predictions when training with energy consumption as label. An alternative to training explicit predictions is to incorporate the learned weights of a speed prediction model into an energy prediction model. This approach involves first training a speed predictor using a DNN, and then incorporating the hidden layers in the energy prediction model. These hidden layers take the same input features as the energy prediction model, and the output of the layers are concatenated to the input of the energy prediction model.

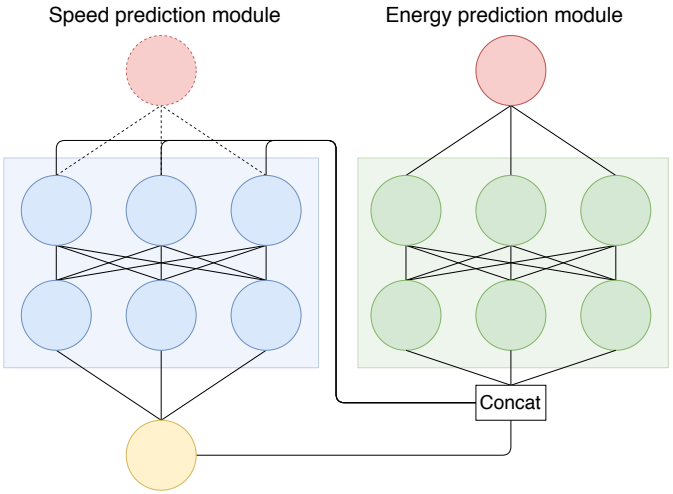


Figure 4.5: Architecture with incorporated speed model.

A visualization of this incorporated speed model can be seen in Figure 4.5. The blue and green rectangles represent the speed prediction and energy prediction modules, respectively. The speed prediction module is trained first, using speed as the label. After this, the output layer

of the speed prediction module is removed (visualized in the figure with dotted lines), and the hidden states of the topmost hidden layer are concatenated to the input of the energy prediction module. The energy prediction module then trains using both the regular input, and the hidden states of the speed prediction module, using energy consumption as the label.

The speed prediction and energy prediction modules can be implemented with many variants of neural networks. We have chosen to implement the speed prediction module as a DNN, and the energy prediction module as an RNN. This means, that after pretraining the speed module, we need to copy the network for every time step in the energy prediction module, in our case by using the TimeDistributed wrapping layer in Keras.

4.4.2 Embedding Categorical Features

As mentioned in Section 2.3, Zhang et al. [43] represents categorical features by embedding them. This approach of embedding categorical features as a dense feature vector is well known within the area of recommender systems, for example implemented by factorization machines. A concrete example of an approach to embedding categorical features for neural networks is given by Zhang et al. [44], who leverage these categorical feature transformation methods to improve the click-through-rate prediction performance of their models.

Inspired by Zhang et al. [43] and Zhang et al. [44], with the aim to better represent categorical features in our model, we implement a variant of our RNN where categorical features are embedded instead of one-hot-encoded. While the model proposed by Zhang et al. [44] is applied to click-through predictions, their approach to learning lower-dimensional real dense feature vectors is still applicable as a method of representing categorical features in our model. In our implementation, visualized in Figure 4.6, we embed each of our categorical features (road category, weekday, and month), as well as the quarter-hour from midnight (quarter) feature. Each of these four features are individually passed through separate dense neural networks, in our case Embedding layers in Keras. The separate networks each receive an one-hot vector as input. This one-hot vector is multiplied by the weight matrix of the network for each category, to obtain the embedding vector for the category specified by the one-hot vector. In Table 4.2 this matrix multiplication is visualized. The one-hot vector has a single element set to one, which acts as a lookup index in the weight matrix, giving us the desired embedding vector.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 0.086 & 0.409 & 0.140 \\ 0.126 & 0.135 & 0.178 \\ 0.192 & 0.050 & 0.053 \\ 0.010 & 0.001 & 0.240 \\ 0.246 & 0.034 & 0.081 \end{bmatrix} = \begin{bmatrix} 0.010 & 0.001 & 0.240 \end{bmatrix}$$

Table 4.2: Result of one-hot lookup in weight matrix.

The resulting feature vectors are concatenated to the rest of the feature set and given as the input for the first hidden layer. It is important to note that Figure 4.6 omits some arrows from the first input layer and the embedding layers to the hidden layer for visualization purposes, and that these are implemented as fully connected layers. By using this idea of limiting each category to separate networks during training, it will also be possible to visualize the differences the neural network learns between the various feature vectors within each category.

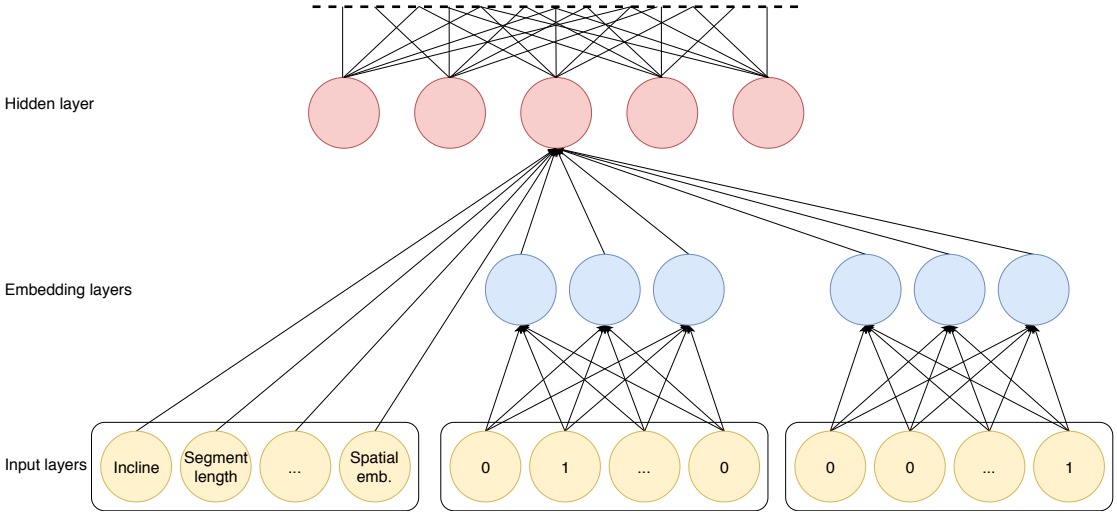


Figure 4.6: Categorical embeddings. Some arrows omitted for visualization purposes.

In the case of the pretrained incorporated speed model introduced in Section 4.4.1, both the energy prediction and speed prediction part of the model include a separate learnable set of embeddings. This is visualized in Figure 4.7. We refer to this RNN with architectural changes as ERNN (Extended Recurrent Neural Network).

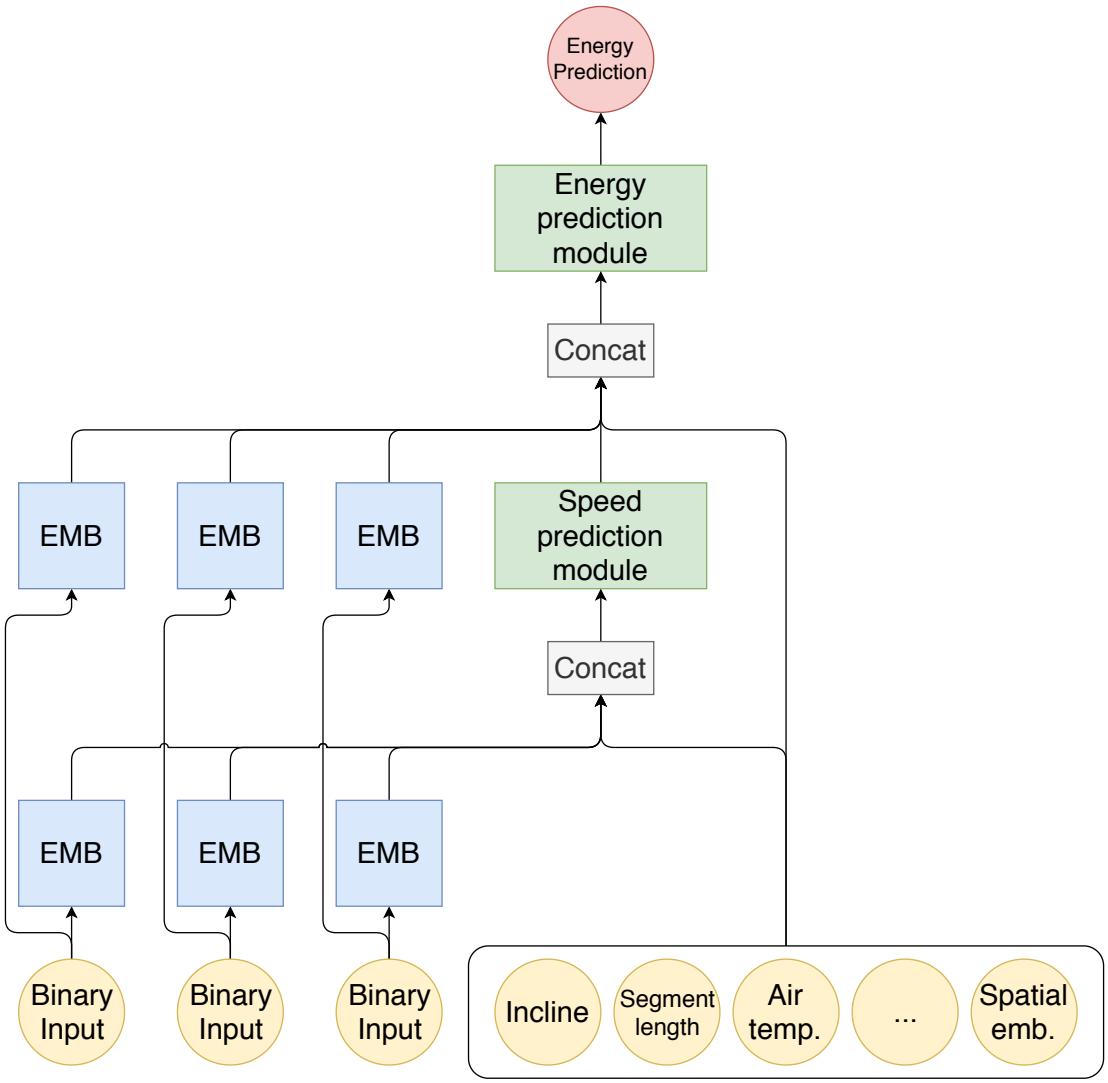


Figure 4.7: Architecture with incorporated speed model and embeddings for categorical features.

Chapter 5

Experiments

In this chapter, we will explain the method by which we optimize the hyperparameters of our models, which results we get from testing them on our dataset, and what we can interpret from the results and from the embeddings of categorical features.

The results are obtained by averaging two consecutive runs, which is necessary due to run-to-run variations caused by our use of NVIDIA GPU libraries with Keras. Running the models on a CPU yields deterministic results, but is one to several orders of magnitude slower depending on the hardware. The tests are evaluated both by their performance on individual segments, using R^2 as the metric, and on the combined trip performance, using trip MAPE. The equation for R^2 can be seen in Equation (5.2), and the equation for MAPE can be seen in Equation (5.1). The reason for using trip MAPE to measure trip performance rather than trip R^2 is because it is more interpretable. For trip segments, MAPE can explode, due to short segments, where the absolute error might be small, but the relative absolute error can be large. For example, on a short segment where the energy consumption is one Watt hour (Wh) and the model predicts 10 Wh, the absolute error is only 9 Wh, but the absolute percentage error is 900%.

$$\text{MAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right| \tag{5.1}$$

$$\begin{aligned} \bar{f} &= \frac{1}{|X|} \sum_{\mathbf{x} \in X} f(\mathbf{x}) \\ SS_{tot} &= \sum_{\mathbf{x} \in X} \left(f(\mathbf{x}) - \bar{f} \right)^2 \\ SS_{res} &= \sum_{\mathbf{x} \in X} \left(f(\mathbf{x}) - f'(\mathbf{x}) \right)^2 \\ R^2 &= 1 - \frac{SS_{res}}{SS_{tot}} \end{aligned} \tag{5.2}$$

5.1 Baseline Label Experiments

As the first step of experimentation, we compare the performance of our baseline when using the newly calculated label, explained in Section 3.6, versus the old label.

The parameters used for this test are the parameters used by the final model from Bundgaard et al. [7]. The parameters are six hidden layers, 1000 cells per layer, ReLU as the activation

function, and the network is trained with the Adamax optimizer for 20 epochs. As mentioned in Section 4.1, a speed model with the same parameters is trained beforehand to obtain speed predictions for all trip segments.

The results obtained with the previous and re-calculated label can be seen in Table 5.1. These results are for the validation dataset, explained in Section 4.1.3, and run with the features specified in Appendix D as input. From the table, we can see a significant improvement in both the R^2 and trip MAPE metrics when using the new label. As was mentioned in Section 4.1, this is due to the new label now properly capturing the energy consumption for a trip segment. All further tests will be conducted with the new label.

Baseline	R^2	Trip MAPE
Old label	0.626	46.550
New label	0.848	33.567

Table 5.1: Performance of the baseline model with the old and new labels.

5.2 Parameter tuning

The first set of experiments we conduct for our four different models is network size experiments. This includes testing the number of hidden layers and number of cells per layer. In the case of our extended RNN, this includes separately testing network size for the speed prediction module, as well as testing the embedding size for the categorical feature embeddings.

The hyperparameters used for these tests are as follows:

- Activation: ReLU
- Optimizer: Adamax
- Hidden Layers: 2-7
- Cells per layer: 250-2000

We first test the range of hidden layers, where we use 1000 cells per layer, which was the number used in the final model from Bundgaard et al. [7]. After adjusting the number of hidden layers, we test the appropriate number of cells per layer, ranging from 250 to 2000, doubling with each step. Following the tuning of the network sizes, we evaluate whether regularization is necessary for all models. Finally, we test whether a context window of five segments performs better than a context window of three.

5.2.1 Baseline

For the baseline model, we conduct a range of experiments, seen in Figure 5.1. These results are for the validation data. For the hidden layer sizes, we see a large increase in R^2 when going above two hidden layers, leveling out at five layers. The trip performance metric (MAPE) is more erratic, not seeing a consistent change between every parameter value. We can see that for the hidden layers, five, six, and seven layers perform equally well on individual trip segments, however, the model with five hidden layers performs better for the trip metric as well.

For the tuning of the number of cells per layer, the metrics are much more consistent. It is evident that reducing the number of cells below 1000 carries a large penalty on both metrics, and increasing it to 2000 sees a very small benefit in MAPE, with a correspondingly small tradeoff in

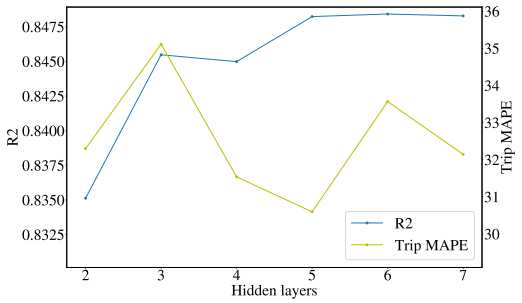
R^2. The difference in R^2 is negligible (0.0006) whereas the increase in MAPE is equally low at 2000 cells per layer, at 0.35 percentage points. Networks with more than 2000 cells per layer take significantly longer to train, and as such we did not test these. Therefore, the chosen number of cells is 1000 per layer. We find that for all sizes, the model converges around 20 epochs, which is consistent with the optimal training time found by Bundgaard et al. [7] for the baseline model.

Following the tests for network size, we evaluate the use of regularization in the baseline model. Bundgaard et al. [7] found that regularization did not improve the model, likely due to the large amount of training data. In Figure 5.1c, the results for our testing of regularization can be seen. We see the same trend with the new label, where increasing the amount of regularization results in a decrease in performance on the validation dataset for the baseline model.

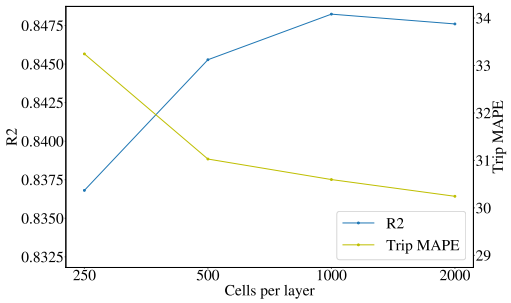
The final baseline results, seen in Table 5.2, are obtained with five hidden layers, 1000 cells per layer, and no regularization.

R^2	Trip MAPE
0.8482	30.5966

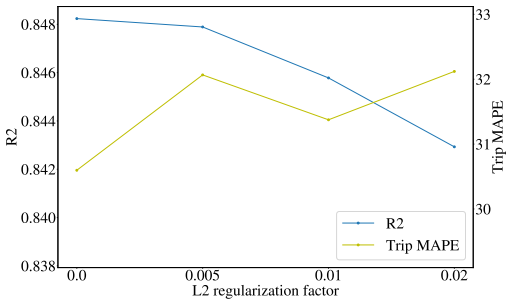
Table 5.2: Result of parameter tuning of the baseline model.



(a) Hidden layer test for the baseline model.



(b) Cells per layer test for the baseline model.



(c) Regularization test for the baseline model.

Figure 5.1

5.2.2 Supersegments

For the supersegment model, we conduct the same range of experiments as for the baseline, seen in Figure 5.2. For this model, changing the number of layers also produces erratic behaviour,

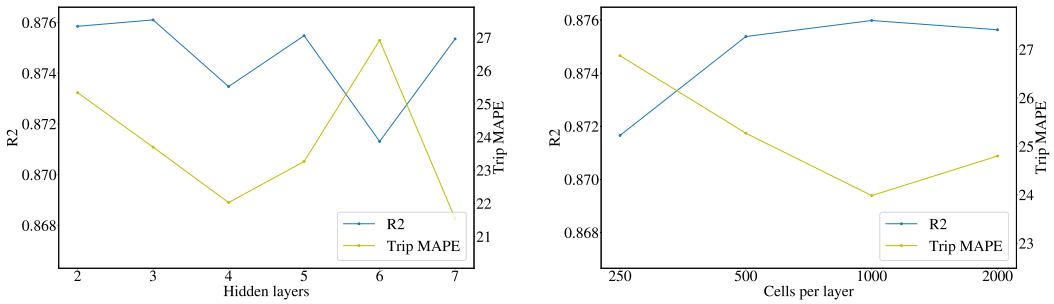
whereas changing the number of cells brings more consistent results. As with the baseline, the supersegment model performs well at five layers, but has consistently better performance at three hidden layers. The parameters chosen here are three hidden layers, with 1000 cells per layer. We then perform experiments with regularization for the supersegment model, with the same result as for the baseline model. Adding regularization strictly decreases the performance on the validation data for this model as seen in Figure 5.2c, similarly to the baseline model.

The final supersegment results for the validation dataset, seen in Table 5.3, are obtained with three hidden layers, 1000 cells per layer, and no regularization. This model performs significantly better than the baseline model in both metrics, indicating that the added context contributes positively to the prediction performance of the model.

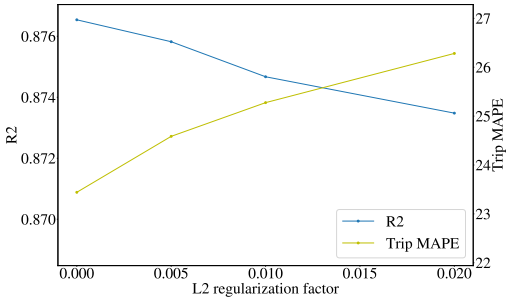
As an additional test, performed using the final parameters achieved above, we evaluate the usefulness of increasing the number of segments in the supersegment model to five in the context window. This does not seem to improve the model compared to the smaller default window size of three segments total, with the best result achieving an R^2 of 0.876 and a trip MAPE of 29.675.

R^2	Trip MAPE
0.8765	23.4412

Table 5.3: Result of parameter tuning of the supersegment model.



(a) Hidden layer test for the supersegment model. (b) Cells per layer test for the supersegment model.



(c) Regularization test for the supersegment model.

Figure 5.2

5.2.3 RNN

For the RNN model, we first test the various network configurations explained in Section 4.3. This includes all four combinations of forwards and reverse many-to-one and many-to-many configurations, as well as a bidirectional many-to-many configuration. We find that the bidirectional many-to-many configuration performs the best, both in terms of R^2 and trip MAPE. This is likely due to the many-to-many configuration being able to learn from all time steps, adjusting the weights based on the loss for both the contextual trip segments, as well as for the focus segment. Bidirectionality also allows the model to capture dependencies in either direction, leading to further improvements.

For this reason, we select the bidirectional many-to-many configuration for further testing, which we conduct the same way as for the supersegment model. Similarly to the supersegment model, three hidden layers and 1000 cells per layer perform well, being the best compromise between segment and trip performance. Additionally, when testing L2 regularization, we also find similar results to the baseline and supersegment models, that no amount of regularization improves the model performance.

The final RNN results, seen in Table 5.4, are obtained with three hidden layers, 1000 cells per layer, and no regularization. For both of the R^2 and trip MAPE metrics, the standard RNN model performs slightly worse than the supersegment model. As was the case for the supersegment model, we test the addition of further segments in the context window for the RNN. This also results in no improvement, with an R^2 of 0.870 and a MAPE of 28.426.

R^2	Trip MAPE
0.8756	27.9392

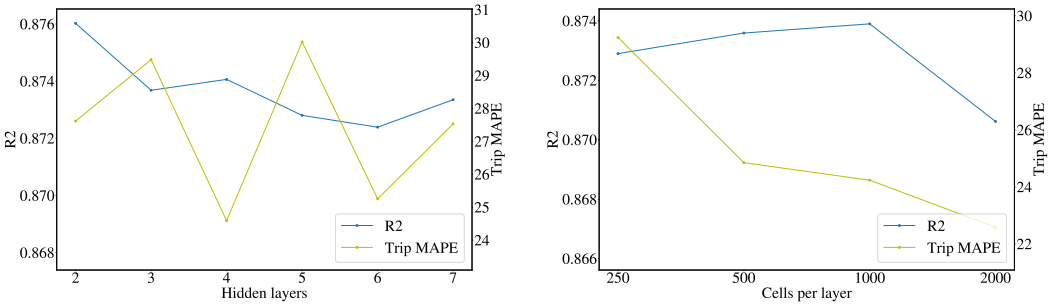
Table 5.4: Result of parameter tuning of the RNN model.

5.2.4 Extended RNN

In Section 4.4, we detailed additions to an RNN architecture for a model that incorporates a pretrained speed model, as well as embeddings of categorical features. For the parameter tuning of this model, we take a different approach to tuning the amount of hidden layers, due to the presence of the speed module. If we first tune the size of the energy prediction module, it may result in the network relying more on the large speed prediction module to learn the intricacies of energy consumption. Therefore, we tune the speed module first, followed by the energy prediction module. The embedding sizes of the categorical features also have to be tuned, which is done following the network size parameter tuning. Finally, we perform testing for L2 regularization as the last step for the ERNN.

We start by adjusting the size of the speed prediction module, followed by tuning the size of the energy prediction module. The speed module is pretrained on the speed label available for trip segments prior to being incorporated in the ERNN, meaning that tuning the module affects its capability to capture the interactions between speed over a segment and the input features. Our tests indicate that the speed prediction module does not need to be large, as increasing the number of hidden layers above three or the number of cells per layer above 250 leads to a drop in both R^2 and an increase in trip MAPE for the energy predictions.

When tuning the size of the energy prediction module, we also find that four layers finds the most improvement for the models' predictive performance, as seen in Figure 5.3a. For the number of cells, 1000 obtains the best results as seen in Figure 5.3b.



(a) Hidden layer test for the ERNN model.

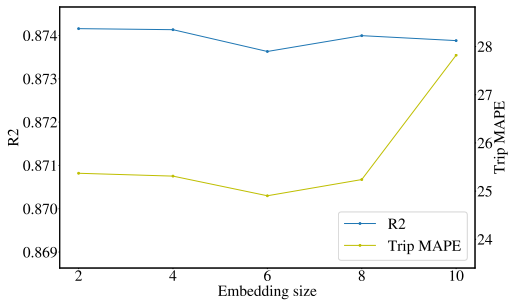
(b) Cells per layer test for the ERNN model.

Figure 5.3

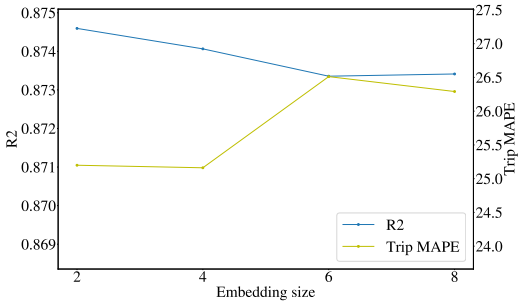
After tuning the size of the ERNN, we adjust the size of the embedding vectors for the categorical features. The size of an embedding vector can be thought of as the number of factors affecting energy consumption that can be captured for each feature. If the interaction between a categorical feature and the energy consumption is complex, it is likely that a larger embedding size is required. For features with simpler interactions or fewer possible values, we would therefore expect a smaller embedding size to be sufficient. When adjusting the embedding sizes, we start with the embeddings for the road category feature, followed by month, weekday, and finally the quarter feature. The tuning of the latter embeddings will use the best parameters found in the previous tests.

In Figure 5.4a, we show the effect of changing the size of the embeddings for the road category feature. It is evident that this feature does not have large variations in per-segment performance (R^2), but it is notable that there is large variation in trip MAPE when changing the size of the vectors. The difference between the worst trip MAPE and the best is almost as large as the variation we saw when tuning the number of hidden layers in Figure 5.3a. This means that the road category embedding is likely a good indicator of trip-level interactions, even if the per-segment performance is almost static. As we mentioned in our analysis of the road category feature in Section 3.1, road category is a likely indicator of the speed that a vehicle will travel at due to road categories having different speed limits. The embedding size we choose for the road category feature is four.

For the month feature, a much lower variance in performance metrics is evident, as seen in Figure 5.4b. This is likely due to the fact that month is primarily an indicator of temperature as shown in our analysis of this feature in Figure 3.8. For this feature, we select the embedding size to be two.



(a) Embedding size test for the road category feature.

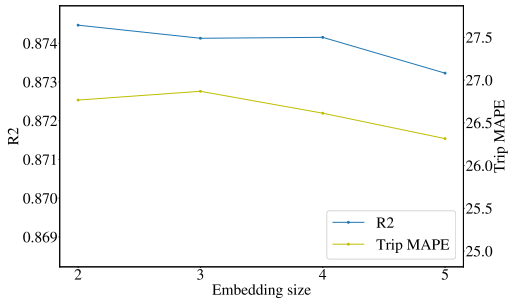


(b) Embedding size test for the month feature.

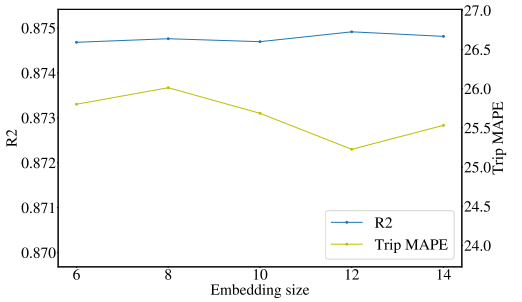
Figure 5.4

For the weekday feature, seen in Figure 5.5a, we observe little variance between the different embedding sizes. The R^2 metric becomes slightly worse as embedding size increases. The trip MAPE improves as the embedding size increases from three, with two performing slightly better than three. Due to the low variance, and an embedding size of two resulting in the best R^2, we continue with this as our embedding size for the weekday feature.

Figure 5.5b shows the results of different embedding sizes for the quarter feature. Here, we again see very little variance between the values, however, the best R^2 and trip MAPE are observed with an embedding size of 12, which we therefore choose as the embedding size for quarter.



(a) Embedding size test for the weekday feature.



(b) Embedding size test for the quarter feature.

Figure 5.5

Following the evaluation of embedding sizes, we test the amount of regularization for the ERNN model. In Figure 5.6, the results for regularization can be seen. We first evaluate the same regularizaton factors as for the other models, being 0, 0.005, 0.01 and 0.02. An immediate result is that the model without regularization performs the best on the trip MAPE metric. However, the model with a regularization factor of 0.005 performs significantly better on individual segments, even though it has an increased trip MAPE. We therefore branch further to include regularization factors of 0.0025 and 0.0075, and find that the best result on individual segments is with a regularization factor of 0.0075. For performance on trips, the model with no regularization applied performs the best. This is in contrast to the results we obtained for the other models, that do not seem to benefit in any way from regularization. We therefore wish to include both

of these variants of the ERNN model in our further testing. The final ERNN results obtained for the validation dataset, seen in Table 5.5, are obtained with a model size of four hidden layers with 1000 cells per layer. The ERNN model also does not improve from an increased context window size. The embedding sizes are:

- 4 for Road Category
- 2 for Month
- 2 for Weekday
- 12 for Quarter

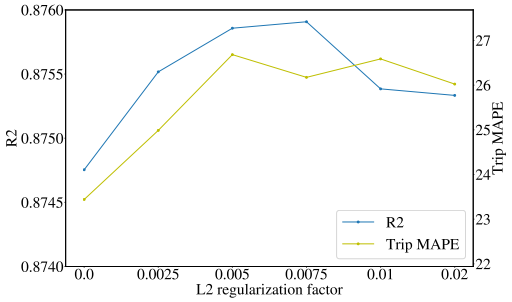


Figure 5.6: Regularization test for the ERNN model.

L2 Regularization	R^2	Trip MAPE
0.0	0.8748	23.4408
0.0075	0.8759	26.1738

Table 5.5: Result of parameter tuning of the ERNN model.

5.2.5 Model Comparison by Trip Length

Motivated by the stark difference in the results from the ERNN on trip MAPE versus R^2 on segments when applying regularization, we perform an analysis to compare trip MAPE for models on groups of differing length of trips. We divide trips into groups of one kilometer increments from zero kilometers to 10 kilometers, with the last group containing all trips longer than 10 kilometers. The results of this comparison can be seen in Figure 5.7. The result for the first group can be seen in Table 5.6, as all models perform significantly worse on this group.

The results for the supersegment model, seen in Table 5.3, indicated that this model performs better in all aspects when compared to the results for the ERNN models in Table 5.5.

When we instead analyze *where* the models perform better, we see that the reason for this significant improvement is due to the supersegment model attaining a good result for the shortest group of trips as seen in Table 5.6. In contrast, the ERNN model with L2 regularization applied performs better on all trips longer than 1 km when compared to the other models, which also explains the variance in the metrics when tuning the hyperparameters.

This is an interesting result, as we are interested in a better performance on longer trips, due to the fact that range anxiety plays a larger role for these trips. For trips that are very short,

say below 1 kilometer, range anxiety is of much less concern as it is unlikely the drivers of these trips will run out of battery.

We also see that some architectures for the same models trade between performing well on short trips, versus performing better on longer trips or vice versa. This is showcased by the parameter tuning results for the supersegment and ERNN models, which have large swings in MAPE when tuning the number of hidden layers (Figure 5.2a and Figure 5.3a).

From Figure 5.7, we can see that the ERNN with L2 regularization performs very well on the other trip length groups, compared to the other models. This suggests that adding L2 regularization guides this model towards a goal that generalizes better on longer trips for the validation data. The same cannot be said for the supersegment architecture, which did not benefit from regularization.

These results therefore indicate that short trips are characterized by different factors than long trips, due to the trade-off in performance. Following the hyperparameter tuning on the validation data, we analyze the contribution of the input features to the performance of the ERNN model.

Model	0-1km MAPE
Baseline	170.231479
RNN	126.565180
ERNN_L2	115.639433
Supersegment	98.232964
ERNN	78.059656

Table 5.6: Model trip MAPE for the 0-1km group.

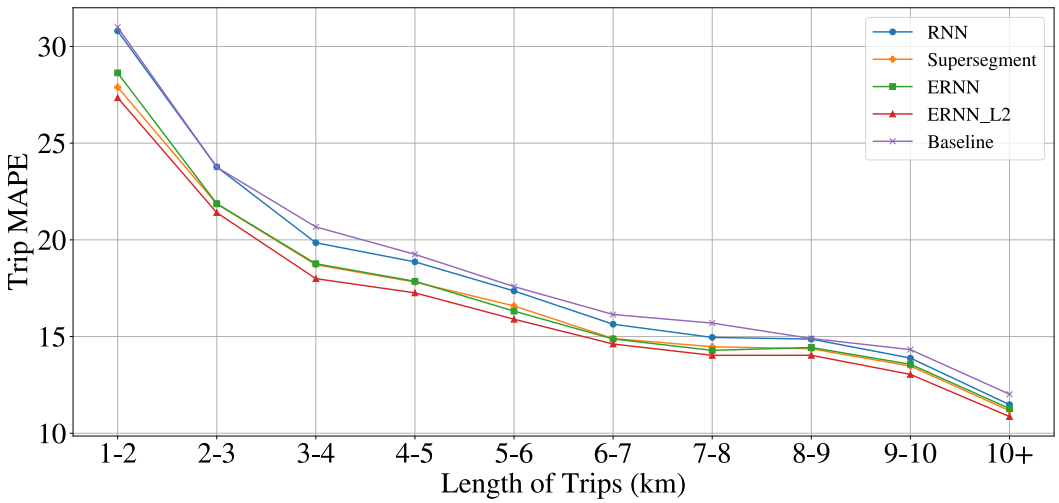


Figure 5.7: Comparison of performance by trip length.

5.2.6 Feature Analysis

In this section, we analyze the impact of the various input features used in the ERNN model. An overview of the impact of each input feature can be seen in Table 5.7. The table is sorted based on the drop in the R^2 metric, with the model that includes all input features at the top. Features towards the bottom of this table provoke the largest drop in performance once excluded. Note that we do not include the categorical features, as we instead choose to analyze the feature vectors learned by the ERNN for these features in the following section. The ERNN model incorporates a speed prediction module, and the contribution of this module to the final result is also measured. The speed prediction module is normally trained on a speed label prior to being incorporated in the ERNN, as explained in Section 4.4. For the test that removes the pretrained speed, we simply do not pretrain the weights of the speed module prior to incorporating it in the ERNN. This means that we keep the same architecture and measure the contribution of the separate training of the speed module.

Of the feature contributions shown in Table 5.7, we see that the features we include in our dataset (described in Chapter 3) have varying degrees of usefulness. These features are the incline, traffic light, roundabout, average speed, and angle features. The angle feature has the largest contribution to model performance of these, followed by the incline. This likely means, as the analysis of the angle feature indicated in Section 3.4, that the angle of a turn between segments is a good indicator of changes in the energy consumption between segments. The positive contribution of the incline feature also corresponds well with our analysis of the correlation between this feature and energy consumption in Section 3.2, which shows a clear correlation between changes in incline and reductions or increases in average energy consumption over sections.

On the other hand, for traffic lights and roundabouts the results are less positive. The exclusion of these features show no significant change in the performance of the model, leading us to believe that their contributions are already captured by some of the other features. This could be a combination of the angle and "degree" features, which identify turning and intersections. The angle feature is only available for models which include other context segments, since it is necessary to know which segments the angles are between.

The historical weather data, namely the temperature and headwind speed features, show positive contributions as well. From our analysis of these features in Section 3.1, our expectation would be that the temperature feature is more indicative of changes in energy consumption. When removing these features, the exclusion of the headwind speed shows a larger decrease in performance. We believe this is caused by the network using the month and other time-related features to compensate for the missing temperature, as the month feature is a primary indicator of temperature.

The speedmap we obtained from Vejdirektoratet, described in Section 3.5, lead us to creating the average speed feature. Our analysis of this feature showed a clear correlation between the actual speed and the predicted speed for the segments which had a measurement from the speedmap. The average speed feature also contributes positively to the performance of the ERNN, meaning that obtaining precise historical speed measurements for all road segments rather than a subset is likely to improve the predictive performance of future energy prediction models.

For the test that excludes pretraining of the speed module, we see that this has a substantial effect on the ERNN model. The pretraining of the weights in the speed module allows the complete energy prediction model to improve significantly, due to it explicitly learning a representation of the speed a vehicle will drive at over a segment.

Removed feature	R^2	Trip MAPE
None	0.8759	26.1738
Traffic light	0.8759	26.9619
Speed limit	0.8757	27.1670
Roundabout	0.8757	24.9982
Degree	0.8753	25.8521
Average speed	0.8746	24.9109
Temperature	0.8744	25.8848
Headwind speed	0.8742	26.3519
Pretrained speed	0.8738	24.4371
Incline	0.8724	28.2329
Angle	0.8709	26.3854
Segment length	0.8601	26.2763

Table 5.7: Result of removing feature, performed with the ERNN model.

In Figure 5.8, we further analyze the contribution of the segment length, angle, and incline features, as well as the pretrained speed component. We show the progression in trip MAPE with regards to the length of trips, allowing us to see where each feature contributes. This is to determine whether these features are primarily indicators of energy consumption for longer or shorter trips.

The exclusion of the angle feature shows that performance on trips of all lengths, which are all characterized by turns, significantly decreases. This is not the case for the pretrained speed component, which primarily contributes more precise predictions for trips that are six kilometers or longer. A likely cause of this result is that the speed module is not perfectly capturing all local interactions, but that these errors cancel out for longer trips and therefore still allow for more precise predictions.

The incline feature is shown to be a strong contributor to model performance at short trip lengths, but sees a reduction for long trips. This is likely due to the fact that the amount of energy spent on climbing hills is a smaller part of trips that have a significant duration.

The length of segments is an important indicator of energy consumption for all trip lengths. This is a natural conclusion to make, as traveling in a vehicle implies performing a certain amount of work to travel a given length.

In the following section, we will analyze the embeddings learned by the ERNN model for the remaining categorical features. These are the embeddings that capture a temporal aspect, namely month, weekday, and quarter, in addition to the road category feature.

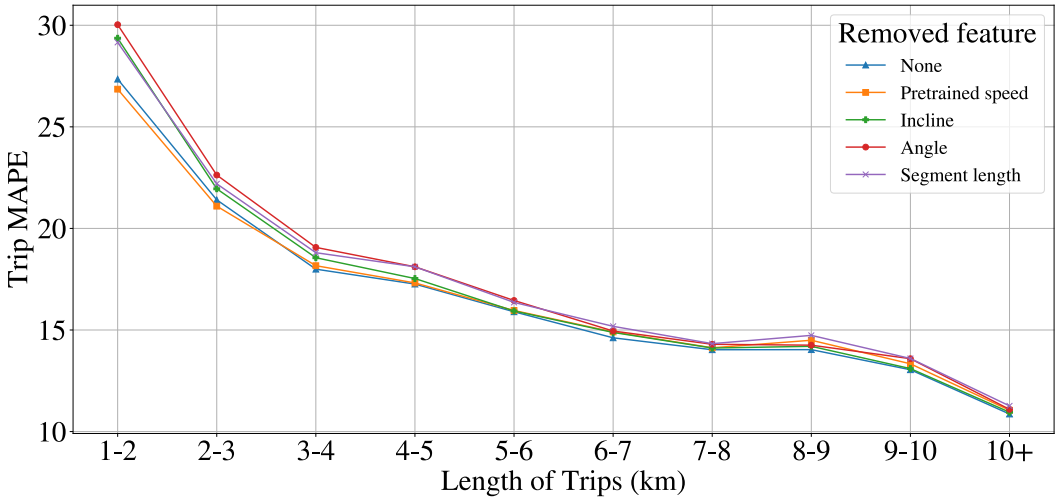


Figure 5.8: Impact on trip MAPE by feature removal.

5.3 Embedding Weights

In the ERNN model, we use small, separate layers with identity activation functions to embed our categorical features as vectors of real numbers. These layers take as input a one-hot encoding of the categorical value and multiply it by a weight matrix, making them linear transformations. This allows us to extract and compare the rows of the weight matrix, each of which directly show how one categorical value is embedded.

We have two different embedding layers, one used as input for the speed prediction module and one for energy prediction, for each of the four categorical features: road category, month, weekday, and quarter. In this section, we will focus on the energy embeddings, and will exclude the quarter feature, as the 96 distinct values makes the embeddings too large to interpret directly. However, all the embedding vectors for both the speed and energy prediction modules can be seen in Appendix G. These vectors are trained with the final ERNN model.

One measure for showing the similarity between the embedding vectors with varying magnitudes is cosine similarity, definable as $\frac{A \cdot B}{\|A\|\|B\|}$ for vectors A and B. It is 1 if the vectors are parallel, 0 if they are orthogonal, and -1 if they are opposite. Thus, assuming that the learned embeddings represent some underlying qualities of the categorical values, higher cosine similarities may show that two values share similar qualities, even if they have different amounts of those qualities. Cosine similarity does not take into account the magnitude of the vectors, but its boundedness makes it easy to visualize, which is why we will use it in the following analysis.

In each of the following tables, categorical values are shown on the left and top, while the intersecting cells contain the cosine similarity of the embedding vectors for the two values. Values have been rounded to two digits for space reasons and are colored in a spectrum going from red for values close to 1, through white for values close to 0, to blue for values close to -1.

5.3.1 Road Category

Our dataset contains 17 different road categories from OSM[31]. These include motorway, trunk, primary, secondary, and tertiary roads and a link category for each, which includes on and off ramps and other roads that carry traffic to and from roads of that category. Of the five, the first four connect cities and larger towns, while tertiary roads primarily connect smaller towns and suburbs. Other road categories are unclassified (minor roads between towns), residential, living streets (shared with pedestrians and bicyclists), service (access roads for buildings, service stations, etc.), track (for agricultural and forestry use), unpaved, and road. The generic road category contains segments for which the classification is unknown. As such, they do not have any commonality, and since only 80 trip segments were driven on the road category, we choose to exclude it from this analysis.

Table 5.8 shows the similarities between the embeddings used as input for the energy model for different road categories. It contains three main clusters with similar embeddings.

	Motorway	Trunk	Primary	Secondary	Motorway link	Trunk link	Primary link	Secondary link	Tertiary	Tertiary link	Unclassified	Residential	Living street	Service	Track	Unpaved
Motorway	1,0	0,0	0,5	0,7	-0,5	-0,8	-0,6	-0,2	0,0	-0,4	-0,6	-0,5	-0,7	-0,9	-0,4	-0,1
Trunk	0,0	1,0	0,7	-0,6	0,0	0,3	0,0	0,2	-0,1	-0,1	-0,4	0,5	0,1	0,3	0,1	0,5
Primary	0,5	0,7	1,0	0,1	0,0	-0,1	0,0	0,4	0,1	-0,1	-0,9	-0,2	-0,6	-0,4	-0,5	0,0
Secondary	0,7	-0,6	0,1	1,0	-0,1	-0,6	-0,2	0,1	0,2	-0,1	-0,5	-0,9	-0,8	-0,9	-0,7	-0,7
Motorway link	-0,5	0,0	0,0	-0,1	1,0	0,8	1,0	0,9	-0,3	0,1	-0,2	-0,2	-0,2	0,1	-0,5	-0,6
Trunk link	-0,8	0,3	-0,1	-0,6	0,8	1,0	0,9	0,7	-0,2	0,2	0,1	0,3	0,3	0,6	0,0	-0,2
Primary link	-0,6	0,0	0,0	-0,2	1,0	0,9	1,0	0,9	-0,1	0,3	-0,2	-0,2	0,0	0,3	-0,4	-0,6
Secondary link	-0,2	0,2	0,4	0,1	0,9	0,7	0,9	1,0	-0,1	0,1	-0,6	-0,4	-0,5	-0,1	-0,8	-0,7
Tertiary	0,0	-0,1	0,1	0,2	-0,3	-0,2	-0,1	-0,1	1,0	0,9	-0,2	-0,4	-0,1	0,0	-0,1	-0,3
Tertiary link	-0,4	-0,1	-0,1	-0,1	0,1	0,2	0,3	0,1	0,9	1,0	-0,1	-0,3	0,1	0,3	0,0	-0,3
Unclassified	-0,6	-0,4	-0,9	-0,5	-0,2	0,1	-0,2	-0,6	-0,2	-0,1	1,0	0,6	0,9	0,6	0,9	0,5
Residential	-0,5	0,5	-0,2	-0,9	-0,2	0,3	-0,2	-0,4	-0,4	-0,3	0,6	1,0	0,8	0,8	0,8	0,9
Living street	-0,7	0,1	-0,6	-0,8	-0,2	0,3	0,0	-0,5	-0,1	0,1	0,9	0,8	1,0	0,9	0,9	0,7
Service	-0,9	0,3	-0,4	-0,9	0,1	0,6	0,3	-0,1	0,0	0,3	0,6	0,8	0,9	1,0	0,7	0,5
Track	-0,4	0,1	-0,5	-0,7	-0,5	0,0	-0,4	-0,8	-0,1	0,0	0,9	0,8	0,9	0,7	1,0	0,8
Unpaved	-0,1	0,5	0,0	-0,7	-0,6	-0,2	-0,6	-0,7	-0,3	-0,3	0,5	0,9	0,7	0,5	0,8	1,0

Table 5.8: Cosine similarity between embeddings of categorical values for the road category feature in the energy model.

The first is the cluster with link roads, all pairs of which have similarities at or above 0.7. This could indicate that the embeddings capture the similar patterns of acceleration and deceleration, and their effects on energy consumption, when driving onto or off from larger roads.

The second is the cluster with tertiary and tertiary link roads. These are mostly orthogonal to other road types, but unlike the other road/link pairs, are very similar to each other. This could indicate that the embeddings capture different driving patterns for roads in more rural areas, and that the road segments linking onto them are seldom separated enough from the tertiary roads to have different driving patterns.

The third is the cluster with unclassified, residential, living street, service, track, and unpaved category roads. Residential, living street, and service roads are mostly present in cities and towns, which makes the similarity understandable. The unclassified category is often applied to road segments between smaller groups of houses in residential areas, which helps explain its inclusion in the cluster. Track and unpaved category roads, on the other hand, are mostly found in rural areas, which makes their inclusion surprising, but there may be underlying qualities that warrant it, for example in the form of a cautious driving style at a lower speed. The similarity in the embedding vectors for the speed model, included in Appendix F, show a similar cluster of the living street, service, track, and unpaved categories, while excluding the unclassified and residential categories, which to some extent supports this interpretation.

5.3.2 Month

Table 5.9 shows the similarities between the embeddings used as input for the energy model for different months. It displays a general, cyclic trend that months that are close to each other are also similar, and that months in different seasons are dissimilar. This indicates that the embeddings capture the seasonal variations in temperature and precipitation which influence battery efficiency, air-conditioning usage, tire grip, driving cautiousness, and speed. It also aligns with the cyclical pattern seen in our analysis month in Section 3.1.1.

The one outlier in the cycle is in between July and August, which show a distinct split. Here, the embedding vectors for the months before and after the split are mostly orthogonal. One interpretation could be a difference in driving conditions during and after the Danish school summer holidays, which begin in late June and end in early August, but it does not explain the lack of a similar split at the beginning of the holidays.

	January	February	March	April	May	June	July	August	September	October	November	December
January	1,0	0,3	-0,3	-0,8	-1,0	-1,0	-1,0	-0,2	0,2	0,5	0,8	0,9
February	0,3	1,0	0,8	0,3	0,0	-0,3	-0,4	-1,0	-0,9	-0,6	-0,4	-0,1
March	-0,3	0,8	1,0	0,8	0,6	0,4	0,2	-0,9	-1,0	-1,0	-0,8	-0,6
April	-0,8	0,3	0,8	1,0	0,9	0,8	0,7	-0,5	-0,7	-0,9	-1,0	-1,0
May	-1,0	0,0	0,6	0,9	1,0	1,0	0,9	-0,2	-0,4	-0,8	-0,9	-1,0
June	-1,0	-0,3	0,4	0,8	1,0	1,0	1,0	0,1	-0,2	-0,6	-0,8	-1,0
July	-1,0	-0,4	0,2	0,7	0,9	1,0	1,0	0,3	0,0	-0,4	-0,7	-0,9
August	-0,2	-1,0	-0,9	-0,5	-0,2	0,1	0,3	1,0	1,0	0,8	0,5	0,2
September	0,2	-0,9	-1,0	-0,7	-0,4	-0,2	0,0	1,0	1,0	0,9	0,7	0,5
October	0,5	-0,6	-1,0	-0,9	-0,8	-0,6	-0,4	0,8	0,9	1,0	0,9	0,8
November	0,8	-0,4	-0,8	-1,0	-0,9	-0,8	-0,7	0,5	0,7	0,9	1,0	1,0
December	0,9	-0,1	-0,6	-1,0	-1,0	-1,0	-0,9	0,2	0,5	0,8	1,0	1,0

Table 5.9: Cosine similarity between embeddings of categorical values for the month feature in the energy model.

5.3.3 Weekday

Table 5.9 shows the similarities between the embeddings used as input for the energy model for different weekdays. It contains two clusters.

The first cluster contains Monday, Tuesday, Wednesday, and Thursday. As these are the four full workdays, this may indicate that the embedding layer learns to capture underlying traffic trends related to rush hour traffic. While Friday is a workday, it is often shorter for many workers, which could cause the dissimilarity with the first four weekdays. The embedding vectors for Tuesday and Wednesday are somewhat orthogonal, which we do not have an explanation for.

The second cluster contains Friday and Saturday. The embeddings may be capturing traffic patterns related to weekend activities, such as lower traffic in residential areas, where people are at home, and higher traffic in cities.

The Sunday embedding vector is somewhat similar to the Friday and Saturday vectors, which could have indicated some shared pattern of energy consumption over the whole weekend. However, the magnitude of the Sunday vector is many orders smaller than all the other vectors. As such, it seems instead that the embedding layer learned that none of the underlying factors it captured for the two clusters apply to Sundays.

Both of these cluster interpretations align with the patterns across weekday and time seen

in Figure 3.10, where the weekend days are slightly different from the workdays especially at midday.

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Monday	1,0	0,8	0,9	1,0	-0,9	-0,9	-0,8
Tuesday	0,8	1,0	0,4	0,8	-1,0	-1,0	-0,2
Wednesday	0,9	0,4	1,0	0,9	-0,6	-0,6	-1,0
Thursday	1,0	0,8	0,9	1,0	-0,9	-0,9	-0,8
Friday	-0,9	-1,0	-0,6	-0,9	1,0	1,0	0,5
Saturday	-0,9	-1,0	-0,6	-0,9	1,0	1,0	0,5
Sunday	-0,8	-0,2	-1,0	-0,8	0,5	0,5	1,0

Table 5.10: Cosine similarity between embeddings of categorical values for the weekday feature in the energy model.

5.4 Evaluation on Test Dataset

Following the parameter tuning of the various models on the validation dataset, we find that a model based on an RNN architecture, the ERNN, performs significantly better than the super-segment model which is implemented as a DNN.

To determine how well the ERNN generalizes, a test dataset containing 30% of the total amount of data has been withheld. We first compare the performance of the baseline against the ERNN, to determine the magnitude of improvement that a model exploiting context information is capable of. This comparison can be seen in Table 5.11. The ERNN model attains a 15.87% lower MAPE metric compared to the baseline model, calculated for all trips. For trips above 10 kilometers in length, the MAPE obtained by the ERNN model is 13.89% lower. A model that incorporates contextual information therefore significantly outperforms a model which does not consider this context.

In Figure 5.9, we show the trip segment performance (in R^2) for each road category on the test dataset. Here, the ERNN model performs best on categories that tend to be part of longer trips such as the motorway and trunk categories. Note that the category "road" has been discarded as it is a road category for roads with unknown categories, as mentioned in Section 5.3.1. The ERNN model performs on par with the baseline in the trunk and motorway road categories, but the gap grows larger for the categories that are not major road arteries. For example, for tertiary roads, which is the most driven category in our dataset, the ERNN improves on the per-segment performance by a much larger margin.

Model	Trip MAPE	Trip MAPE >10 km
Baseline	22.31	12.38
ERNN	**18.77**	**10.66**

Table 5.11: Model performance on test set for the baseline and ERNN.

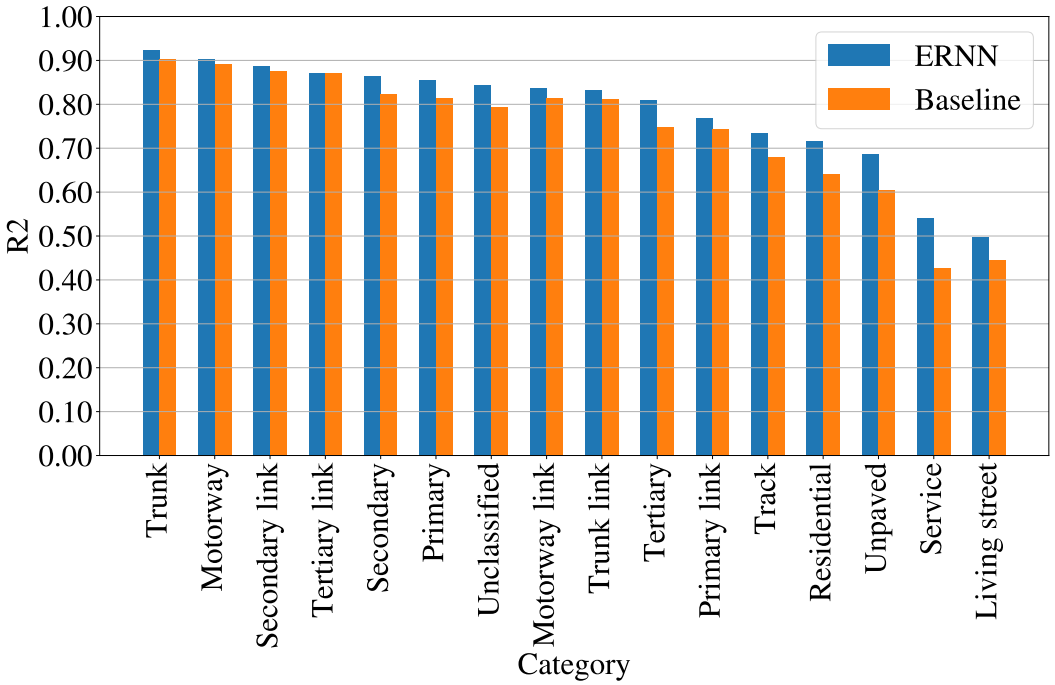


Figure 5.9: Performance on trip segments grouped by category, scored with R^2.

In Table 5.12, we show the performance by trip length of the ERRN model on the validation dataset as well as on the test dataset. It is evident that the model generalizes quite well, and manages to keep similar MAPE across both the validation and the test datasets. A surprising result is that the model performs significantly better on the shorter trips in the test dataset. This is perhaps caused by some outlier trips in the validation data, but is not investigated further. The MAPE for the very long trips (>10 km) decreased for the test set, meaning that the model still performs very well for these trips.

Trip length	Validation	Test
0-1 km	115.639	44.968
1-2 km	27.348	26.675
2-3 km	21.406	21.131
3-4 km	17.994	18.224
4-5 km	17.261	16.914
5-6 km	15.897	15.359
6-7 km	14.615	14.951
7-8 km	14.029	14.102
8-9 km	14.033	13.603
9-10 km	13.044	13.507
>10 km	10.864	10.661

Table 5.12: MAPE by trip length for the ERNN model on the test and validation data.

In Table 5.13, we show the MAPE for three further categories, which contain trips above 10 km in length. It shows a promising result, which is that the model performs better as trips increase in length. A MAPE of 10% indicates that the model is on average 10% off with its predictions. There is a natural ceiling that limits the predictive performance of models that do not take into account the specific driving parameters of the driver of the vehicle. Research for conventional vehicles show that there can be large differences in fuel consumption for different drivers of similar vehicles[19]. This can be due to various factors, primarily related to whether the driver follows eco-driving advice such as maintaining speed and driving at or below the speed limit.

Since our model does not take into account any information related to who is driving the vehicle, this naturally limits how precise it will be able predict the energy consumption for arbitrary trips performed by many different individuals. As such, we believe that a MAPE of 10.6% for trips above 10 kilometers in length is a very positive result. It shows that data-driven methods that take into account the many factors that influence energy consumption, including trip context, can realistically attain results that can inform electric vehicle drivers about the range potential of their vehicles.

Trip length	MAPE
10-20 km	11.276
20-30km	9.814
>30 km	9.802

Table 5.13: MAPE by trip length over 10 km for the ERNN model.

Chapter 6

Conclusion

In this chapter, we will summarize our contributions and, by that, attempt to answer the questions of our problem statement:

- Is it possible to accurately predict energy consumption for electric vehicles given the complex non-linear interactions between the factors in our dataset by using neural networks?.
  - Is it possible to exploit the sequential nature of trip data to improve energy predictions?
  - Can data about the intersections a trip goes through improve the energy predictions?
  - How does different handling of categorical features influence the energy prediction?
  - How can the prediction model help understand underlying patterns in the data?

In this project, we extended the energy prediction framework proposed by Bundgaard et al. [7] to incorporate contextual information from trips. We implemented a baseline model without context directly based on their work for comparison. Then, we implemented three different models which exploit the described context for improved energy predictions: a supersegment model, an RNN model, and an ERNN model. We showed how all of the models improved on the baseline, with the ERNN model achieving a mean absolute percentage error (MAPE) of 18.77% for all trips and 10.66% for trips over 10 km. This is a significant improvement on the baseline of 22.31% for all trips and 12.38% for trips over 10 km. The improved performance on longer trips is especially relevant, where the limits to battery size may induce range anxiety which accurate predictions can help alleviate.

Measuring the mean absolute error (MAE) as a percentage of the average energy consumption of a trip, the ERNN model achieves a score of 12.24%. This compares favorably to the neural network model proposed by De Cauwer et al. [10], which achieves an MAE of 12 to 14% of the average energy consumption of a trip in their datasets.

By improving the performance over the baseline in both our RNN-based models, we showed that it is possible to exploit the sequential nature of trip data via a model architecture specialized for sequential data. However, by showing a similar, if lesser, improvement in our simpler supersegment model, we also showed that such a specialized architecture isn't a necessity to exploit it. This may inform other projects on sequential data with very local context, where a simpler DNN-model is preferred.

As input for our models, we engineered several new features related to road intersections. These include turn angles, number of roads meeting at intersections (degree), roundabouts, and traffic lights. During our ablation tests, we showed that the inclusion of turn angle and degree

improved energy predictions, while information about roundabouts and traffic lights did not. This proves the usefulness of extracting data about intersections from the road network and the raw geometries describing it. It also suggests that resources might be spent better in other ways than by mapping roundabouts and traffic light if the model is to be implemented for other geographical areas where this information is not already available.

Finally, for the ERNN model, we implemented embedding of four categorical features which were included in one-hot encoded form in the other models. The four features are the road category each trip segment was driven on and the month, weekday, and quarter-hour it was driven at. We did not directly compare the influence of this change separately from the other change in the ERNN model, which was the inclusion of a pretrained module for speed prediction. However, the ERNN model showed significant improvements in prediction performance compared to the RNN model.

Additionally, adding the categorical embedding layers allowed us to extract and compare the learned embedding vectors for each categorical value. This provided insights into the energy consumption patterns the model learned for these categories, which would otherwise have been hard to extract from the deep neural network models. Other projects which want to increase explainability of DNNs may benefit from the same technique.

6.1 Future Work

We see future work within this area related to the following topics:

Investigating the applicability of battery state features in an energy prediction model. These features are not available in our dataset, and obtaining health status of batteries in the electric vehicles could lead to the ability to compensate for aging batteries with reduced capacity.

An alternative route is to consider the integration of driver profiles in an energy prediction model. Our data foundation explicitly excludes user information, but it is still relevant to consider whether this prevents grouping drivers according to certain traits visible from existing factors in the dataset.

Finally, investigating the capability of the data-driven models to generalize to other areas is a possible option. A concrete example of this could be attempting to create accurate predictions for a dataset with more and different vehicle models of varying weight classes.

Bibliography

[1] O. Andersen, B. B. Krogh, and K. Torp. Analyse af elbilers forbrug. In *Trafikdage på Aalborg Universitet (2014)*, 2014.

[2] O. Andersen, B. B. Krogh, and K. Torp. Analyse af elbilers forbrug for perioden 2012-2013. Technical report, Daisy, Institut for Datalogi, Aalborg Universitet, 2014.

[3] O. Andersen, B. B. Krogh, and K. Torp. Elvis: Comparing electric and conventional vehicle energy consumption and co2 emissions. In M. Gertz, M. Renz, X. Zhou, E. Hoel, W.-S. Ku, A. Voisard, C. Zhang, H. Chen, L. Tang, Y. Huang, C.-T. Lu, and S. Ravada, editors, *Advances in Spatial and Temporal Databases*, pages 421–426, Cham, 2017. Springer International Publishing.

[4] B. Asadi and A. Vahidi. Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time. *IEEE transactions on control systems technology*, 19(3):707–714, 2011.

[5] M. Baum, J. Dibbelt, T. Pajor, and D. Wagner. Energy-optimal routes for electric vehicles. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 54–63, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2521-9. doi: 10.1145/2525314.2525361.

[6] M. Baum, J. Dibbelt, A. Gemsa, D. Wagner, and T. Zündorf. Shortest feasible paths with charging stops for battery electric vehicles. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '15, pages 44:1–44:10, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3967-4. doi: 10.1145/2820783.2820826.

[7] J. Bundgaard, H. Hansen, R. Jensen, B. Madsen, C. Nørskov, and L. Pipiras. An embedding-based approach to predicting energy consumption of electric vehicles. `https://projekter.aau.dk/projekter/files/294234942/mi904e18_project_report.pdf`, 2019. Semester project for our 9th semester.

[8] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[9] C. De Cauwer, J. Van Mierlo, and T. Coosemans. Energy consumption prediction for electric vehicles based on real-world data. *Energies*, 8(8):8573–8593, 2015. ISSN 1996-1073. doi: 10.3390/en8088573. URL `http://www.mdpi.com/1996-1073/8/8/8573`.

[10] C. De Cauwer, W. Verbeke, T. Coosemans, S. Faid, and J. Van Mierlo. A data-driven method for energy consumption prediction and energy-efficient routing of electric vehicles in real-world conditions. *Energies*, 10(5):608, 2017.

[11] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[12] GDAL. Gdal - geospatial data abstraction library. `https://www.gdal.org/`, 2019. Accessed: 2019-06-11.

[13] Geofabrik. Openstreetmap data for denmark. `https://download.geofabrik.de/europe/denmark.html`, 20140101. Accessed: 2019-06-11.

[14] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[15] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[16] S. Hochreiter and J. Schmidhuber. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997.

[17] Hyundai Danmark. Hyundai KONA electric - selvsikker og enestående, 2018. URL `https://www.hyundai.dk/kona-electric`.

[18] International Energy Agency. Global EV Outlook 2018, 2018. URL `https://webstore.iea.org/global-ev-outlook-2018`.

[19] K. Jakobsen, S. C. H. Mouritsen, and K. Torp. Evaluating eco-driving advice using gps/-canbus data. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 44–53, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2521-9. doi: 10.1145/2525314.2525358.

[20] S. Kanarachos, J. Mathew, and M. E. Fitzpatrick. Instantaneous vehicle fuel consumption estimation using smartphones and recurrent neural networks. *Expert Systems with Applications*, 120:436 – 447, 2019. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.12.006.

[21] Kortforsyningen. Dhm-2007/terræn_bro. `https://download.kortforsyningen.dk/content/dhm-2007terr%C3%A6nbro`, 2007. Accessed: 2019-06-11.

[22] B. Krogh, O. Andersen, and K. Torp. Analyzing electric vehicle energy consumption using very large data sets. In *International Conference on Database Systems for Advanced Applications*, pages 471–487. Springer, 2015.

[23] J. Leskovec and R. Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

[24] Mitsubishi Motors. Mitsubishi i-miev specification. `https://www.mitsubishi-motors.com/en/showroom/i-miev/specifications/`, 2011. Accessed: 2019-06-11.

[25] Z. Mohamed-Kassim and A. Filippone. Fuel savings on a heavy vehicle via aerodynamic drag reduction. *Transportation Research Part D: Transport and Environment*, 15(5):275 – 284, 2010. ISSN 1361-9209. doi: https://doi.org/10.1016/j.trd.2010.02.010.

[26] National Climatic Data Center USA. Nndc climate data online, 2019. URL `https://www7.ncdc.noaa.gov/CDO/country`.

[27] J. Neubauer and E. Wood. The impact of range anxiety and home, workplace, and public charging infrastructure on simulated battery electric vehicle lifetime utility. *Journal of Power Sources*, 257:12 – 20, 2014. ISSN 0378-7753. doi: https://doi.org/10.1016/j.jpowsour.2014.01.075.

[28] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 336–343. ACM, 2009.

[29] Nissan. New nissan leaf - range & charging, 2018. URL `https://www.nissan.dk/biler/personbiler/leaf/distance-opladning.html`.

[30] OpenStreetMap. Openstreetmap aalborg background tile. `https://www.openstreetmap.org/#map=19/57.00981/9.98857`, 2019.

[31] OpenStreetMap Community. Key:highway, Feb 2019. URL `https://wiki.openstreetmap.org/wiki/Key:highway#Roads`.

[32] PostGIS. Postgis - spatial and geographic objects for postgresql. `https://postgis.net/`, 2019. Accessed: 2019-06-11.

[33] QGIS. Qgis - a free and open source geographic information system. `https://www.qgis.org`, 2019. Accessed: 2019-06-11.

[34] Tesla, Inc. Tesla: Model 3, 2018. URL `https://www.tesla.com/da_DK/model3`.

[35] The Danish Energy Agency. Test en elbil. `https://ens.dk/ansvarsomraader/transport/alternative-drivmidler/forsoegsordning-elbiler`, 2010-2014.

[36] T. Tielert, M. Killat, H. Hartenstein, R. Luz, S. Hausberger, and T. Benz. The impact of traffic-light-to-vehicle communication on fuel consumption and emissions. In *2010 Internet of Things (IOT)*, pages 1–8. IEEE, 2010.

[37] M. van der Steen, R. M. Van Schelven, R. Kotter, M. J. W. van Twist, and P. van Deventer MPA. *EV Policy Compared: An International Comparison of Governments' Policy Strategy Towards E-Mobility*, pages 27–53. Springer International Publishing, Cham, 2015. ISBN 978-3-319-13194-8. doi: 10.1007/978-3-319-13194-8_2. URL `https://doi.org/10.1007/978-3-319-13194-8_2`.

[38] B. O. Varga, A. Sagoian, and F. Mariasiu. Prediction of electric vehicle range: A comprehensive review of current issues and challenges. *Energies*, 12(5):946, 2019.

[39] Vejdirektoratet. Grundlag for udformning af trafikarealer - anlæg og planlægning. `http://vejregler.lovportaler.dk/ShowDoc.aspx?docId=vd20180154-full`, 2018. Accessed: 2019-06-11.

[40] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang. Modeling trajectories with recurrent neural networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3083–3090, 2017. doi: 10.24963/ijcai.2017/430.

[41] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[42] T. Yuksel and J. Michalek. Effects of regional temperature on electric vehicle efficiency, range, and emissions in the united states. *Environmental science & technology*, 49, 02 2015. doi: 10.1021/es505621s.

[43] H. Zhang, H. Wu, W. Sun, and B. Zheng. Deeptravel: a neural network based travel time estimation model with auxiliary supervision. *CoRR*, abs/1802.02147, 2018. URL `http://arxiv.org/abs/1802.02147`.

[44] W. Zhang, T. Du, and J. Wang. Deep learning over multi-field categorical data. In *European conference on information retrieval*, pages 45–57. Springer, 2016.

[45] B. Zheng, P. He, L. Zhao, and H. Li. A hybrid machine learning model for range estimation of electric vehicles. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2016.

Appendix A

Traffic Light SQL Queries

```
1  CREATE VIEW trafficlight_points AS
2  SELECT DISTINCT point FROM (
3      WITH segments AS (
4          SELECT osm.startpoint, osm.endpoint, trf.light_id
5          FROM (SELECT osm_id as light_id, ST_Transform(way, 4326)::
               ↪ geography AS light_geog
6                FROM osm_archive_point
7                WHERE highway = 'traffic_signals') AS trf
8          JOIN (SELECT startpoint, endpoint, segmentgeo
9                FROM osm_map) AS osm
10         ON ST_DWithin(trf.light_geom, osm.segmentgeo, 20)
11     )
12
13     SELECT startpoint as point, light_id
14     FROM segments
15
16     UNION ALL
17
18     SELECT endpoint as point, light_id
19     FROM segments
20 ) points
21 GROUP BY light_id, point
22 HAVING count(1) > 1
```

Code Example A.1: SQL query for obtaining segment start- and endpoints that are traffic lights.

```
1  SELECT startpoints.segmentkey,
2         CASE WHEN startpoints.startpoint IS NOT NULL
3              THEN 1
4              ELSE 0
5         END AS startpoint_is_trafficlight,
6         CASE WHEN endpoints.endpoint IS NOT NULL
7              THEN 1
8              ELSE 0
9         END AS endpoint_is_trafficlight
10 FROM (SELECT segmentkey, point AS startpoint
11        FROM osm_map AS osm
12        LEFT JOIN trafficlight_points AS tfl
13        ON tfl.point = osm.startpoint) AS startpoints
14 JOIN (SELECT segmentkey, point AS endpoint
15        FROM osm_map AS osm
16        LEFT JOIN trafficlight_points AS tfl
17        ON tfl.point = osm.endpoint) AS endpoints
18 ON startpoints.segmentkey = endpoints.segmentkey
```

Code Example A.2: SQL query for obtaining segment start- and endpoints that are traffic lights.

Appendix B

Roundabout SQL Queries

```
1  SELECT DISTINCT osm.segmentkey,
2         CASE WHEN bcj.junction = 'roundabout'
3              THEN 1
4              ELSE 0
5         END AS is_roundabout
6  FROM bcj_line AS bcj
7  RIGHT JOIN osm_map AS osm ON bcj.osm_id = osm.segmentid
```

Code Example B.1: SQL query for determining which road segments are in a roundabout.

```
1  SELECT DISTINCT point
2  FROM(
3      WITH roundabouts AS (
4           SELECT startpoint, ST_Startpoint(segmentgeo::geometry) AS
                 ↪ startpointgeo, endpoint,
5           ST_Endpoint(segmentgeo::geometry) AS endpointgeo
6           FROM bcj_roundabouts AS rab
7           JOIN osm_map AS osm ON rab.segmentkey = osm.segmentkey
8           WHERE is_roundabout = 1)
9
10     SELECT startpoint AS point
11     FROM roundabouts
12
13     UNION ALL
14
15     SELECT endpoint AS point
16     FROM roundabouts
17 ) points
18 GROUP BY point
19 HAVING count(1) > 1
```

Code Example B.2: SQL query for determining which start- and endpoints belong to a roundabout

```
1  SELECT startpoints.segmentkey,
2         CASE WHEN startpoints.startpoint IS NOT NULL
3              THEN 1
4              ELSE 0
5         END AS startpoint_is_roundabout,
6         CASE WHEN endpoints.endpoint IS NOT NULL
7              THEN 1
8              ELSE 0
9         END AS endpoint_is_roundabout
10 FROM (SELECT segmentkey, point AS startpoint
11        FROM osm_map AS osm
12        LEFT JOIN bcj_roundabout_points AS rap
13        ON rap.point = osm.startpoint) AS startpoints
14 JOIN (SELECT segmentkey, point AS endpoint
15        FROM osm_map AS osm
16        LEFT JOIN bcj_roundabout_points AS rap
17        ON rap.point = osm.endpoint) AS endpoints
18 ON startpoints.segmentkey = endpoints.segmentkey
```

Code Example B.3: SQL query for marking start- and endpoints with the presence of a roundabout.

Appendix C

Energy Label Recalculation Query

```
1 SELECT m.id, SUM(ev_watt) / 3600000 AS ev_kwh_from_ev_watt
2 FROM mapmatched_data.viterbi_match_osm_dk_20140101 AS trip_data
3 JOIN gpsdata.factgpsdata AS gpsdata ON gpsdata.id = ANY(trip_data.
    ↪ gpsdata_ids)
4 WHERE ev_watt IS NOT NULL AND ev_watt <= 50000 AND ev_watt >= -20000
5 GROUP BY trip_data.id
```

Code Example C.1: SQL query for recalculating ev_kwh from the GPS points.

Appendix D

Features Used by Baseline

Feature name	Type	Example Value	Description
Category	Categorical	"Residential"	Category of the road segment
Incline	Float	4.32	Percentage incline of the road segment, according to driving direction
Segment length	Float	142.4	Length of the road segment in meters
Temperature	Float	5.3	Temperature in degrees celsius
Headwind speed	Float	2.5	Speed of the headwind in meters per second
Quarters from midnight	Int	95	Number of 15-minute increments from midnight
Weekday	Categorical	"Wednesday"	The weekday
Month	Categorical	"December"	The month
Speedlimit	Int	50	The speedlimit of the road segment
Degree (start/end)	Int	3	Number of connected road segments at the start and end of a road segment
Trafficlight (start/end)	Binary	1	Indication of traffic light at the start and end of a road segment
Roundabout (start/end)	Binary	0	Indication of roundabout at the start and end of a road segment
Average speed from speedmap	Float	43.45	Average speed over the road segment, from Vejdirektoratet's speedmap
Speed prediction	Float	14.5	Speed in meters per second, predicted for a trip segment by a model trained on a speed label
Spatial embedding	Float vector	[0.54, 0.32, ..., -0.75]	Feature vector describing a road segment, obtained with node2vec

Table D.1: Features used by the baseline model.

Appendix E

Preprocessing Pipeline

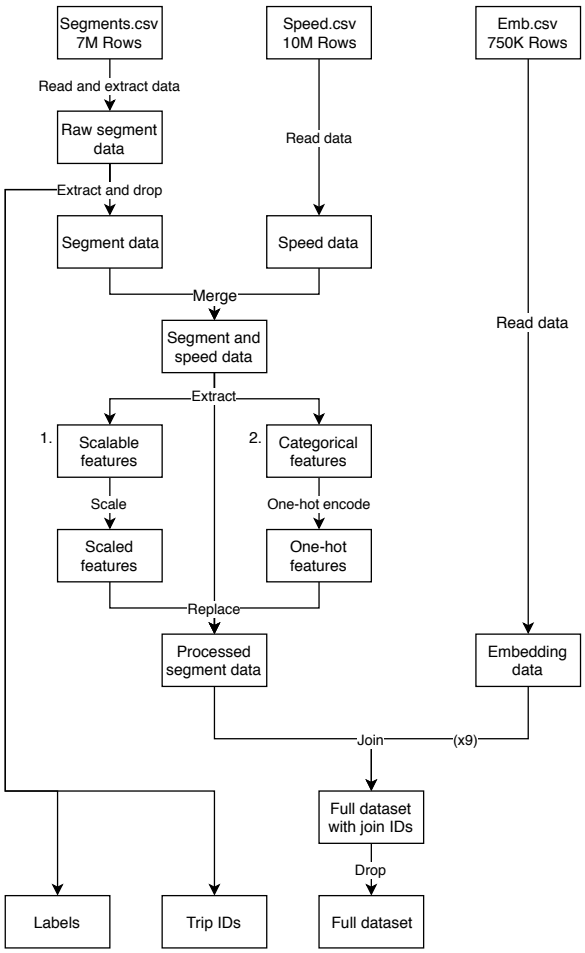


Figure E.1: Data preprocessing pipeline. Notice that the two processes indicated by 1. and 2. are conducted sequentially.

Appendix F

Road Category Embeddings in Speed Model

	Motorway	Trunk	Primary	Secondary	Motorway link	Trunk link	Primary link	Secondary link	Tertiary	Tertiary link	Unclassified	Residential	Living street	Service	Track	Unpaved
Motorway	1,0	0,5	0,2	0,0	-0,2	-1,0	-0,2	0,7	0,2	0,9	0,0	0,1	-0,9	-0,9	-0,9	-0,9
Trunk	0,5	1,0	0,9	0,9	-0,6	-0,6	0,2	0,9	0,0	0,8	-0,6	-0,3	-0,6	-0,5	-0,6	-0,5
Primary	0,2	0,9	1,0	1,0	-0,8	-0,3	0,1	0,8	-0,1	0,5	-0,5	-0,4	-0,4	-0,3	-0,4	-0,3
Secondary	0,0	0,9	1,0	1,0	-0,8	-0,1	0,2	0,7	-0,3	0,4	-0,5	-0,3	-0,2	-0,1	-0,2	-0,1
Motorway link	-0,2	-0,6	-0,8	-0,8	1,0	0,2	0,3	-0,5	0,7	-0,3	0,0	-0,1	0,4	0,4	0,1	0,3
Trunk link	-1,0	-0,6	-0,3	-0,1	0,2	1,0	0,3	-0,8	-0,3	-1,0	0,0	0,1	1,0	1,0	1,0	1,0
Primary link	-0,2	0,2	0,1	0,2	0,3	0,3	1,0	0,0	0,1	-0,1	-0,9	0,2	0,5	0,5	0,3	0,5
Secondary link	0,7	0,9	0,8	0,7	-0,5	-0,8	0,0	1,0	0,2	0,9	-0,5	-0,3	-0,8	-0,7	-0,8	-0,8
Tertiary	0,2	0,0	-0,1	-0,3	0,7	-0,3	0,1	0,2	1,0	0,3	-0,2	-0,7	-0,2	-0,1	-0,5	-0,3
Tertiary link	0,9	0,8	0,5	0,4	-0,3	-1,0	-0,1	0,9	0,3	1,0	-0,3	-0,2	-0,9	-0,9	-0,9	-0,9
Unclassified	0,0	-0,6	-0,5	-0,5	0,0	0,0	-0,9	-0,5	-0,2	-0,3	1,0	0,2	-0,2	-0,3	0,0	-0,2
Residential	0,1	-0,3	-0,4	-0,3	-0,1	0,1	0,2	-0,3	-0,7	-0,2	0,2	1,0	0,2	0,1	0,4	0,2
Living street	-0,9	-0,6	-0,4	-0,2	0,4	1,0	0,5	-0,8	-0,2	-0,9	-0,2	0,2	1,0	1,0	0,9	1,0
Service	-0,9	-0,5	-0,3	-0,1	0,4	1,0	0,5	-0,7	-0,1	-0,9	-0,3	0,1	1,0	1,0	0,9	1,0
Track	-0,9	-0,6	-0,4	-0,2	0,1	1,0	0,3	-0,8	-0,5	-0,9	0,0	0,4	0,9	0,9	1,0	1,0
Unpaved	-0,9	-0,5	-0,3	-0,1	0,3	1,0	0,5	-0,8	-0,3	-0,9	-0,2	0,2	1,0	1,0	1,0	1,0

Table F.1: Cosine similarity between embeddings of categorical values for the road category feature in the speed model.

Appendix G

Embedding Vectors

Category	0	1	2	3
motorway	-0,11446	-0,22555	-0,08546	-0,01023
motorway_link	0,46452	0,297055	0,141629	0,92178
trunk	-0,10562	-0,06351	0,345038	0,015626
trunk_link	0,199953	0,295446	0,250172	0,264309
primary	-0,0586	-0,28158	0,214638	0,082764
primary_link	0,182159	0,101124	0,078066	0,21232
secondary	0,021536	-0,14155	-0,15987	0,041686
secondary_link	0,050383	-0,01686	0,036718	0,107642
tertiary	0,224545	-0,10983	0,01006	-0,16753
tertiary_link	0,033346	-0,00183	0,007516	-0,01432
unclassified	-0,08126	0,458325	-0,12606	-0,22842
residential	-0,20319	0,24985	0,195834	-0,1153
living_street	-0,0121	0,106319	0,02529	-0,05438
service	0,031998	0,192471	0,132936	-0,06294
road	0,006906	0,031179	0,012083	0,015951
track	-0,0595	0,131046	0,033539	-0,14282
unpaved	-0,21928	0,099787	0,14686	-0,19732

Table G.1: Weight matrix for road category embedding layer in energy module.

Category	0	1
January	0,348622	0,266494
February	-0,12722	0,36453
March	-0,3263	0,224406
April	-0,46184	0,004678
May	-0,41654	-0,15233
June	-0,31434	-0,22017
July	-0,2134	-0,21582
August	0,203042	-0,36616
September	0,312434	-0,29804
October	0,368257	-0,14008
November	0,455921	-0,00997
December	0,576518	0,167254

Table G.2: Weight matrix for month embedding layer in energy module.

Category	0	1
Monday	-0,07668	0,22459
Tuesday	-0,17785	0,113087
Wednesday	0,063817	0,361944
Thursday	-0,06473	0,207891
Friday	0,399874	-0,42279
Saturday	0,492079	-0,55038
Sunday	-1,4E-34	-3,8E-34

Table G.3: Weight matrix for weekday embedding layer in energy module.

Category	0	1	2	3	4	5	6	7	8	9	10	11
00:00-00:15	-0,9446	-0,733	1,36877	-1,2919	2,62969	-0,2851	0,24249	-0,3143	-1,7124	3,03151	-1,5548	-1,6136
00:15-00:30	0,81413	3,90361	1,63412	-3,5038	-1,0392	-3,6533	0,82204	-0,8859	-2,3505	3,76815	0,20948	-2,014
00:30-00:45	2,51405	2,11684	3,13766	-0,3128	2,71114	-2,4671	-0,9459	-1,7581	2,04295	-3,1723	2,70901	-0,7914
00:45-01:00	-3,8284	0,15561	2,72508	-3,2366	3,06011	0,80627	-0,9334	-1,4474	1,97725	-0,9098	1,29767	-1,4158
01:00-01:15	-1,8375	0,0651	3,91136	0,46091	-0,2485	0,38441	1,97523	-0,2386	-3,0092	1,34389	0,16971	2,67472
01:15-01:30	-2,2268	-1,8653	3,69163	-3,372	-1,2489	-2,6319	0,2313	-1,2147	1,10186	-0,2887	3,30468	-2,8119
01:30-01:45	-0,2415	1,31217	2,08982	0,87482	-0,2374	-3,8497	-1,1351	0,29769	-1,6689	3,04402	0,84793	-0,7395
01:45-02:00	1,26361	-1,463	1,29177	-1,6987	-0,9443	0,70791	-0,0833	3,28891	-1,0506	1,84967	0,88615	-3,1708
02:00-02:15	-3,5287	-2,0862	2,6982	1,08177	3,29293	0,58139	-2,5447	-3,6624	1,76298	-3,1337	0,73113	0,20733
02:15-02:30	2,43213	-3,3587	-1,3872	-0,4696	3,29541	3,76692	-2,2462	-2,4025	3,30934	-0,9141	-1,1083	1,7833
02:30-02:45	3,2758	0,83769	-2,9839	-0,0662	1E+27	1,77853	0,11514	-3,902	2,86798	1,40102	-0,1483	0,14691
02:45-03:00	0,37322	-1,7584	-0,8089	1,98418	-3,8532	2,02327	-1,182	-2,8991	-2,1917	2,62883	0,0346	-2,2326
03:00-03:15	-2,1011	-1,1947	-0,5362	-0,3575	-1,1185	1,28317	-1,3775	1,99534	-1,9868	2,21814	-3,823	1,55619
03:15-03:30	3,7474	1,18711	-3,2911	-0,2137	3,78757	-0,9247	-2,0446	-0,297	1,3401	-0,4249	1,87169	-3,7646
03:30-03:45	-1,1522	-1,7824	1,9609	-2,1135	-3,0947	-2,2258	-1,3959	-2,1046	2,14005	0,01976	-0,3686	-0,2636
03:45-04:00	1,18027	-1,3873	-1,7171	-0,1856	-0,1447	2,77514	0,00826	2,22763	4,1E+28	1,17307	3,62889	-0,0304
04:00-04:15	0,20521	-2,9829	-0,0265	-1,0869	0,90505	-2,9864	1,3374	-0,8231	1,56868	-1,7747	1,29816	-1,6412
04:15-04:30	2,00449	2,07686	-0,6361	-1,4648	1,73894	-2,7459	0,37515	-2,3596	-0,5732	0,77152	-1,677	2,32767
04:30-04:45	2,25123	-0,7511	-0,0287	3,39078	-1,3095	-1,7585	2,3602	3,10871	-1,6953	1,49198	-2,2181	-1,5792
04:45-05:00	0,89307	2,83927	-2,3319	0,72062	3,24608	-2,1105	-0,1319	-2,1003	-1,3808	2,31578	2,72163	2,47563
05:00-05:15	-1,6388	-0,0377	1,58793	0,69361	0,13473	3,25066	3,30118	1,78178	1,02565	0,91273	-3,7291	3,52502
05:15-05:30	-1,0985	3,50101	2,85465	-2,2243	-3,53	0,89376	-2,5728	-3,4137	1,32597	-0,1382	-2,1919	3,64283
05:30-05:45	-3,5688	-1,1081	-2,6385	3,56236	2,48106	0,86138	-2,7441	-1,7314	3,77185	-2,6347	1,40966	-3,1682
05:45-06:00	-0,7861	-3,0687	-3,1508	-1,0069	-2,7853	-1,1124	-1,3134	0,49127	2,45305	-1,5287	-1,8946	3,42373
06:00-06:15	1,02107	2,22826	-1,1813	0,08173	0,21034	-1,6754	2,86553	0,55486	1,66859	-1,6674	-0,2351	2,48473
06:15-06:30	-1,1498	-1,8924	-3,1374	-1E+25	-2,8755	-4E+22	-0,6231	1,55395	-3,7661	0,9367	-2,062	-1,1936
06:30-06:45	-1,8783	1,29276	-2,7215	2,49152	-0,937	-2,1393	-2,0951	2,21016	-1,3867	-3,5374	-0,9205	-0,842
06:45-07:00	-0,0446	2,72992	-2,3413	0,28986	2,3392	-1,0292	-3,1709	1,73954	0,9877	0,86837	-3,3744	-0,0805
07:00-07:15	3,32805	0,0884	-1,1829	1,16692	3,66414	1,85245	-0,2425	3,22647	-3,1106	1,81716	-0,7363	-1,1255
07:15-07:30	1,24042	-2,9836	-2,6251	-1,1457	-1,4201	-0,8539	-3,3869	2,45864	-0,281	2,15687	0,11831	2,57184
07:30-07:45	-1,7747	1,19003	1,35906	-0,49	2,08213	-2,5639	-1,0509	-3,4346	-2,309	-0,834	-1,8184	1,82588
07:45-08:00	-3,4568	-0,4812	3,23296	3,49335	-1,6616	1,31983	-0,6348	-2,6915	3,31536	-1,786	-1,6277	-1,5526
08:00-08:15	-0,6874	0,00097	-2,3839	2,19337	3,42413	2,34671	1,88707	-1,8582	-0,5549	-2,7671	0,1413	2,85592
08:15-08:30	-0,8149	1,83985	2,15781	-0,2046	0,69049	2,25185	3,2671	0,8662	1,21253	-1,3742	1,72645	-0,0355
08:30-08:45	-2,5056	-0,4136	-2,0063	-3,6647	-3,6797	-3,1159	3,56945	0,80474	1,14504	-3,5767	-2,6874	0,94993
08:45-09:00	1,54989	1,3144	-3,1271	-0,8143	0,37909	-0,1537	-0,1449	-1E+24	3,75451	-3,735	3,04513	2,75354
09:00-09:15	-1,4609	2,82236	1,86573	-3,2149	-0,9007	-1,4227	0,91802	-1,0049	2,84107	0,62365	-2,3323	-2,8317
09:15-09:30	-3,1229	3,01581	0,2539	-1,8886	-0,7475	-0,14	-0,1682	-2,351	-3,8496	-2,6116	0,88034	1,67856
09:30-09:45	3,75736	-1,8981	-3,0574	-0,6242	3,48979	1,78668	-1,7509	-1,8656	-3,7015	2,88069	3,49352	-2,3121
09:45-10:00	-0,9574	2,90806	-3,508	0,00209	-2,7823	-3,0729	-0,869	-0,8757	1,94744	-3,6399	0,94739	-3,1898
10:00-10:15	2,72303	3,00563	3,46292	-1,9348	-0,5986	3,7777	-1,4538	-1,3216	-2,0283	2,58419	-1,2519	-2,0973
10:15-10:30	-0,4205	0,50847	-2,0541	0,86888	1,0969	-1,7737	1,9361	1,92679	-2,7657	-0,7473	-0,4869	3,65896
10:30-10:45	-1,5591	0,82595	2,48226	1,24393	3,31477	1,04279	-2,6153	0,69429	-0,4079	3,15319	-3,1452	2,26589
10:45-11:00	0,59702	0,49697	-1,5275	-1,584	3,37954	3,3106	0,00192	-2,7274	-2,7353	1,97959	3,57695	-3,7927
11:00-11:15	-3,0102	-1,5121	-0,662	-2,6014	2,47984	3,59513	3,2E+27	-0,5127	2,88035	-2,3215	-1,6422	2,74019
11:15-11:30	3,10803	3,21068	2,2376	2,32172	-0,556	0,43975	1,69613	0,32822	2,23671	-0,3667	0,35697	1,49084
11:30-11:45	2,93436	-1,0291	2,07589	-0,1435	-1,8459	-2,6067	-0,7961	-3,3871	3,51622	1,29892	2,94697	-1,8509
11:45-12:00	2,33336	0,25835	-2,7747	-1,7227	2,30376	-3,2157	2,44166	-0,5224	1,71976	0,98801	1,76954	3,65414

Table G.4: Weight matrix for quarter-hour embedding layer in energy module. All values multiplied by 10^{34} for readability.

Category	0	1	2	3	4	5	6	7	8	9	10	11
12:00-12:15	3,31491	3,88482	3,55297	2,70883	-0,9252	3,86423	-1,2123	2,07996	-1,5193	-2,2809	3,64248	-0,6445
12:15-12:30	-2,6599	-1,084	-2,1932	3,16384	3,10772	-1,9773	-2,3827	3,39597	-0,7136	-1,6274	2,3367	2,92789
12:30-12:45	0,46334	-2,6546	0,53055	-2,9135	-0,8236	2,40433	-0,9813	0,00665	-0,3664	-0,1399	-3,1726	3,35899
12:45-13:00	-1,9818	1,73828	-2,2131	-2,5766	0,9708	-0,8204	3,46951	2,31015	2,756	-1,4903	2,53889	0,01246
13:00-13:15	-2,88	3,67672	-3,1801	3,15831	0,28208	0,55425	-2,8474	-1,0175	3,87258	3,05823	2,78936	2,45622
13:15-13:30	-1,7415	1,21354	-0,4759	0,50472	0,10477	0,96967	1,95385	2,94433	-2,7114	-0,0073	2,55377	-2,1502
13:30-13:45	1,99502	0,59134	2,84069	0,73838	2,86886	-3,4581	1,97273	3,32682	1,17283	-1,883	0,67039	-2,3946
13:45-14:00	0,8549	-0,5458	-3,2895	-3,5448	-3,0817	0,0049	-3,2666	1,33202	2,27727	3,10677	-0,1854	-0,941
14:00-14:15	0,72773	1,05968	2,22786	-1,4631	-3,2252	-2,3583	2,47788	3,65236	3,6915	1,0403	-1,0317	2,13921
14:15-14:30	-0,8974	2,5896	-1,5769	1,17486	-1,4391	-2,8116	0,6745	-2,581	1,89469	-0,1943	1,54647	-0,9374
14:30-14:45	5E+28	1,997	0,61121	-2,4395	-0,2389	0,94976	-1,5736	1,94736	-1,1506	-0,815	2,32328	3,51477
14:45-15:00	0,17403	-0,5694	-1,5602	2,71804	-3,5084	-0,4791	-0,6525	2,75575	-0,4227	2,50151	-0,0744	-1,918
15:00-15:15	-2,2715	-0,4422	3,05763	-0,0417	-0,6311	-0,6983	0,00685	2,76189	2,00716	1,18843	2,67767	1,83257
15:15-15:30	-2,4993	-2,5628	-0,9344	0,67143	-0,5699	-1,6776	1,33839	0,31856	3,22457	-2,3923	-2,0268	2,64126
15:30-15:45	1,10043	-3,7335	-0,0563	-0,1497	2,06493	0,61302	-3,8423	1,63285	-2,8753	1,89926	-2,0311	1,84833
15:45-16:00	-1,3417	-3,8795	-2,6395	-3,3412	-0,4939	1,75503	2,61787	1,1382	2,51393	2,16742	-0,9381	0,76849
16:00-16:15	1,42272	-2,4498	2,07028	3,02806	-0,2033	-0,0628	-0,4566	3,27582	-2,5036	3,0667	-0,6767	2,26221
16:15-16:30	2,33505	-1,135	-1,6912	2,43071	1,88196	3,73349	-0,8111	1,84136	2,24338	1,61775	0,82791	0,83516
16:30-16:45	-2,3906	-2,5533	1,91486	0,99868	1,53111	-1,8884	2,92299	3,3587	0,51247	-3,2196	-1,217	-0,9041
16:45-17:00	0,23045	1,71839	-2,4157	2,53904	-0,8021	-0,0739	-0,4049	1,099	-2,9625	-1,9126	-2,6101	3,61159
17:00-17:15	-1,0712	-0,047	-2,2815	1,20943	-0,0904	-1,7573	3,04935	-0,5044	2,8661	2,77021	-0,1624	3,58902
17:15-17:30	-3,653	-0,4679	-1,3452	3,48425	3,31459	-0,2513	0,66456	-1,4348	2,18972	0,55289	1,40215	-3,0877
17:30-17:45	-1,7185	-0,381	1,85078	2,6091	-2,6396	-2,0079	-3,0382	-0,9936	-0,0419	-1,2137	-0,8713	-2,3872
17:45-18:00	0,4578	0,22801	2,56311	-3,4879	0,94836	-0,691	2,27876	-2,5898	-0,2063	-0,5109	3,29203	-0,8777
18:00-18:15	0,22394	1,76795	2,83336	2,25157	-3,1032	-1,6171	3,85706	-2,5063	-0,3255	2,71038	3,22238	-0,378
18:15-18:30	0,77483	-1,9025	-3,2418	2,90861	-2,6552	1,2318	0,70949	2,33014	-3,1711	-2,8796	-2,674	-3,4809
18:30-18:45	-3,6484	-2,868	-1,0876	1,94017	-2,5758	0,13263	2,27587	-3,2766	-1,0804	-3,055	2,98843	-2,2219
18:45-19:00	2,05932	2,55254	0,67243	-2,3684	1,80673	1,6186	-1,1134	-3,058	-1,2818	2,71444	3,37118	0,74868
19:00-19:15	0,87665	-2,9519	-1,1557	3,06686	-0,8567	1,04073	-0,0193	-2,705	0,43935	3,21199	-0,4669	1,30934
19:15-19:30	2,92358	-1,9706	-1,4252	1,54928	2,23315	-1,1995	-1,5041	1,86156	-0,1058	-1,978	2,1493	2,31088
19:30-19:45	-0,7665	-3,892	-0,0094	1,787	-0,5766	3,76176	-1,1739	-2,0418	1,48249	-0,5839	-0,0588	1,6223
19:45-20:00	2,97632	-2,0493	-2,7591	-0,4113	-0,9309	-0,696	-0,3739	-0,0514	-0,7939	-2,6566	0,20546	1,17321
20:00-20:15	-2,3618	-3,0529	0,04544	-0,3943	1,13685	-3,1465	-2,8359	0,99174	-0,7102	-1,8406	1,73421	-0,1974
20:15-20:30	-2,261	1,89482	-1,5965	-1,9868	-3,0819	-0,7331	-1,8786	0,64579	-0,6776	2,38028	2,92421	-0,4666
20:30-20:45	-1,3607	-2,4547	-3,7678	3,10896	-1,4106	0,14889	-2,323	1,25722	2,47175	2,4139	1,71525	-0,7485
20:45-21:00	1,00615	-1,1952	-2,5133	2,21241	1,37076	-3,232	-1,171	-3,6879	-2,647	3,75382	-2,0639	0,0248
21:00-21:15	0,10812	3,4117	-3,6953	-0,4066	2,56523	1,09529	-2,822	-1,8168	2,14936	1,43701	-3,4954	-0,6116
21:15-21:30	-2,6581	-1,4499	2,8723	3,19269	-1,1238	-0,5331	0,11214	-0,9624	-1,1649	-2,319	3,02652	3,44434
21:30-21:45	3,15059	-1,0267	3,05946	0,11462	-1,6695	0,60674	-1,0598	-1,4188	3,30195	0,99119	0,50125	-0,5208
21:45-22:00	-2,7726	1,48746	4E+29	2,24544	-1,0142	-0,175	0,94753	3,01119	-3,6747	0,8693	3,78493	1,54996
22:00-22:15	3,05304	3,25641	-0,9917	-1,7721	2,42177	-1,0985	-0,0383	0,8373	0,34273	3,21844	3,38325	0,38699
22:15-22:30	-0,333	2,94333	0,56775	-1,5184	-0,7976	-0,2291	-2,9671	2,71872	1,22365	2,80339	1,72281	0,91858
22:30-22:45	1,95045	2,21977	1,4158	0,87095	-2,1184	-3,4074	0,36973	0,57747	-2,8964	-1,8527	-2,0754	-3,3287
22:45-23:00	8,4E+24	0,65896	-1,4747	3,8209	-1,6121	-0,544	2,5792	3,20835	-2,1206	3,32565	2,66774	-0,0712
23:00-23:15	-0,373	-3,7117	3,49631	-2,5752	-0,8587	-2,2815	-1,421	3,1866	-3,8925	1,61945	-2,3928	0,10229
23:15-23:30	3,68365	1,83508	1,24455	1,09536	0,05628	2,38361	-2,7975	-0,0141	-3,3097	3,91417	-0,6981	-1,4912
23:30-23:45	3,89901	3,61604	3,03357	-2,2188	-1,2986	1,93697	-1,345	0,00758	0,96556	2,39053	-2,149	0,01723
23:45-24:00	-3,6844	-2,1528	3,6996	0,6988	0,30545	0,6733	-2,4514	2,91591	1,02249	0,35231	3,45136	-0,129

Table G.4: Weight matrix for quarter-hour embedding layer in energy module. All values multiplied by 10^{34} for readability. (cont.)

Category	0	1	2	3
motorway	0,166176	0,043333	-0,38874	0,044079
motorway_link	-0,1231	0,194233	0,007371	-0,00147
trunk	-0,00882	-0,14667	-0,12039	0,079264
trunk_link	-0,11216	-0,01058	0,198566	-0,08307
primary	0,010427	-0,40037	-0,10777	0,196898
primary_link	-0,05097	-0,01413	-0,01455	-0,02002
secondary	-0,00821	-0,54594	-0,06098	0,161744
secondary_link	0,005838	-0,0223	-0,0357	0,020991
tertiary	-0,1859	0,262028	-0,13252	0,315809
tertiary_link	0,019428	-0,01314	-0,06603	0,027396
unclassified	0,387973	0,245594	0,20899	-0,01831
residential	0,11634	0,020416	-0,07233	-0,50302
living_street	-0,15947	0,008604	0,169492	-0,12257
service	-0,33971	-0,00246	0,326558	-0,15205
road	-0,00051	0,004829	-0,00968	0,001797
track	-0,0859	-0,0222	0,193528	-0,1639
unpaved	-0,20602	-0,01951	0,253055	-0,16551

Table G.5: Weight matrix for road category embedding layer in speed module.

Category	0	1
January	0,056413	0,161671
February	0,257512	0,096292
March	0,050419	0,106508
April	-0,34328	-0,41074
May	-0,25335	-0,22689
June	-0,34773	-0,29902
July	-0,13128	0,254098
August	-0,13193	0,247723
September	-0,04902	-0,07356
October	0,23016	0,178245
November	0,269422	-0,20945
December	0,301085	0,274706

Table G.6: Weight matrix for month embedding layer in speed module.

Category	0	1
Monday	-0,18082	0,22065
Tuesday	-0,27102	-0,09808
Wednesday	0,088225	-0,1218
Thursday	-0,0198	-0,16563
Friday	0,351318	-0,34804
Saturday	0,393647	0,162715
Sunday	-2,4E-34	-3,4E-34

Table G.7: Weight matrix for weekday embedding layer in speed module.

Category	0	1	2	3	4	5	6	7	8	9	10	11
00:00-00:15	3,07275	1,87095	2,84086	0,69713	3,58254	-0,9145	0,04352	-0,1218	-3,1164	0,89903	1,88184	2,44586
00:15-00:30	-3,848	-2,8969	-0,1632	2,86816	3,71838	-2,3411	3,081	0,62233	-1,0815	-2,7233	-1,5057	-3,8027
00:30-00:45	-0,9342	-1,7438	-1,1567	0,47153	0,70981	0,52381	-0,0379	1,64806	0,45242	-2,0016	-0,0599	2,63207
00:45-01:00	-2,9329	-0,1161	1,00526	1,54274	-0,2588	-3,0363	2,62539	-3,2552	-0,2254	0,38865	-0,8781	1,86922
01:00-01:15	1,73144	-0,2874	3,70686	-1,2562	2,49125	-0,1576	1,90637	-1,6232	-0,0722	1,0763	2,26784	-3,064
01:15-01:30	-0,0052	-2,9707	-0,6568	-2,5512	0,66358	-3,3385	-2,6159	-0,4522	2,99902	-1,3611	0,86749	2,28647
01:30-01:45	0,88439	2,0913	2,85534	0,68809	1,8803	-1,1285	0,71124	3,44243	-0,7203	2,11042	1,02462	1,80963
01:45-02:00	-0,2715	-3,0109	1,42423	-1,6325	0,58515	-2,3933	-0,0092	0,13273	-0,6553	1,44741	1,99101	-2,6033
02:00-02:15	2,13648	-2,9886	-1,2593	-2,4358	-2,2928	-3,0329	1,80809	1,77239	3,17523	1,28569	0,57345	-2,9471
02:15-02:30	-3,5757	-0,1574	2,26455	2,97951	1,20064	-1,7475	-2,9344	1,14877	-2,5712	3,60933	-0,2249	1,32179
02:30-02:45	-1,8318	-1,7064	-3,2937	1,63536	2,774	-1,1714	-2,1355	-2,9015	-1,8711	2,48407	-1,3674	2,66075
02:45-03:00	2,26668	-0,1298	-0,3214	-3,682	-0,9766	2,09504	3,09288	3,25786	3,27539	-1,1369	-2,1046	0,78963
03:00-03:15	0,63683	3,46772	-2,9777	-0,9542	2,79055	-3,2622	0,69655	-3,0789	-0,5696	-2,71	-1,0462	1,38301
03:15-03:30	-1,2322	3,48478	3,02989	-3,1757	3,43499	-1,6685	1,74603	-1,4572	-1,8784	-0,046	-3,0249	0,62151
03:30-03:45	-1,8923	-0,06	0,24614	1,21273	1,53084	0,24827	-0,7656	-2,7423	0,38414	-0,4152	-3,0318	2,59295
03:45-04:00	3,45058	-0,2716	-1,7012	-0,5478	-0,8848	0,29584	1,43547	2,82899	1,17457	-0,7846	-1,6692	0,97598
04:00-04:15	2,93715	1,24434	-3,0907	-0,8878	-0,8806	-0,0203	-1,7956	-2,2347	0,69367	-1,8894	-0,2957	-0,2095
04:15-04:30	-3,8348	-1,3053	-0,3482	1,03484	3,53555	-3,401	-3,904	1,48491	-0,9827	-2,2642	0,0453	-2,2332
04:30-04:45	-3,101	3,56683	-3,2708	-1,3265	-0,1044	2,45303	-2,8075	-1,6509	2,61988	1,41606	2,35963	-0,0501
04:45-05:00	3,35533	3,60506	-0,7457	-2,7314	3,01656	2,91016	2,80909	-1,1406	-1,4641	0,10802	-0,4689	-2,0508
05:00-05:15	3,31365	-2,32	2,27865	3,27114	0,7079	2,07973	-1,2406	1,99934	2,09759	0,99415	0,76733	3,69749
05:15-05:30	-0,7299	1,82515	0,97265	-1,3914	-0,9375	-2,405	-2,4926	1,6758	-3,3094	2,11578	1,75691	-1,3844
05:30-05:45	-2,8768	2,76397	1,17351	2,80515	1,14156	0,56813	-1,6937	0,95212	0,61462	2,87144	-0,6413	1,17898
05:45-06:00	-1,2205	-0,7068	0,40469	-2,6456	2,53358	1,28264	1,54855	1,69511	1,78437	-1,049	2,60325	-2,103
06:00-06:15	3,43158	-2,301	3,34	-2,7085	-0,4196	2,38646	-0,0788	-1,0152	0,89269	3,59193	3,54543	1,76847
06:15-06:30	0,84846	-1,6534	3,1475	1,26311	-1,5914	1,43898	3,25958	0,3038	1,60675	1,4091	2,26154	-1,1781
06:30-06:45	1,95531	-1,287	-1,6638	2,34191	-2,8779	-1,5239	-1,6771	0,25785	-1,4035	2,68033	-1,5243	-0,8583
06:45-07:00	2,2859	2,24375	2,58582	1,3092	3,03428	3,78545	-0,8394	-3,0425	-1,1744	-2,2802	2,14456	-1,5093
07:00-07:15	-2,5076	0,71921	2,69973	3,7648	-1,6377	-3,1962	-3,025	0,68253	1,12268	-3,245	1,00171	-0,3192
07:15-07:30	-1,256	-2,5879	-2,4131	0,92516	3,07312	1,26316	-2,481	-3,1188	-0,2343	-2,256	-2,5665	1,80972
07:30-07:45	-1,7992	-0,544	2,54168	2,76001	-1,9514	-3,0128	-3,5735	-2,441	0,84206	0,9203	2,13509	-2,7002
07:45-08:00	1,57698	-0,5888	0,10355	-0,6637	-1,6936	0,58183	-2,9193	-0,0758	-0,1256	0,33989	-2,691	-2,6452
08:00-08:15	-0,2439	-1,0567	-2,4267	1,02026	0,36786	-0,338	0,24405	0,07985	1,90791	0,2296	-2,5516	0,86107
08:15-08:30	-2,5736	-2,5279	-2,8465	-0,7815	-2,9018	-1,822	1,18874	-3,3248	-1,3097	-2,0845	-1,1746	-2,5694
08:30-08:45	2,48688	-2,9683	2,34164	-0,5789	2,16654	0,97868	-2,592	-3,4207	2,47165	-0,2441	2,35559	3,52318
08:45-09:00	-2,1251	0,34429	0,77329	-0,6904	-0,9936	-3,1686	2,24739	-2,7928	-1,3956	1,19223	-3,1394	2,38149
09:00-09:15	0,75436	0,72409	1,82803	2,43339	-3,6308	1,17851	1,57759	1,45054	0,56048	3,15526	0,08714	-2,4636
09:15-09:30	-2,8425	-2,017	0,68823	-1,6146	3,21419	1,37013	-0,7396	0,67062	-0,7285	0,23063	-1,5984	2,25456
09:30-09:45	-0,6921	2,11118	2,09027	-1,1626	-0,9144	-2,6127	0,98156	0,00794	0,97501	-2,6758	0,40392	-2,3521
09:45-10:00	-0,7551	2,62531	-3,764	1,63463	-2,1761	1,19304	-2,5635	-0,2519	0,94601	3,80595	3,79709	-0,0577
10:00-10:15	3,01137	0,55853	3,8497	-2,7573	-0,7258	-3,212	0,95829	0,32171	1,71115	1,08323	-0,2845	-1,8669
10:15-10:30	-2,4847	3,26435	0,41164	2,68415	-3,7828	3,02467	-1,4167	3,5311	-1,6911	1,30405	1,03039	-0,2283
10:30-10:45	-2,0006	3,60491	-0,4008	0,14022	-2,6666	0,93187	-1,3997	3,82989	-0,4541	2,74285	0,02333	2,92323
10:45-11:00	2,96726	3,3785	0,68764	-0,8847	-0,614	3,85543	1,69391	1,53803	-2,5743	-3,0902	-0,0083	2,41203
11:00-11:15	0,24227	-3,1677	-0,0885	-0,81	-1,8732	-0,4202	2,62548	-0,5892	-0,2221	-1,2359	-1,9918	-2,3683
11:15-11:30	-0,239	-1,9995	1,04903	-1,7614	3,59388	2,55786	-3,1987	3,5876	-0,4639	3,47849	1,65926	1,25052
11:30-11:45	-2,1784	0,14918	-2,6969	-1,7764	-1,0081	0,609	0,52536	1,15474	2,36019	1,5806	-1,745	-3,2707
11:45-12:00	1,32532	3,42086	3,11858	0,79463	2,57765	0,29837	-3,6412	-2,571	-0,7051	-2,3823	0,8671	-2,4697

Table G.8: Weight matrix for quarter-hour embedding layer in speed module. All values multiplied by 10^{34} for readability.

Category	0	1	2	3	4	5	6	7	8	9	10	11
12:00-12:15	1,47625	-2,3207	1,69333	3,10373	1,0596	0,18385	-3,1817	1,21433	0,56022	2,24609	0,14425	-3,7681
12:15-12:30	1,70597	3,64696	-3,0744	0,77082	1,82287	-3,3669	1,36048	1,31407	-2,3311	-3,2861	-3,7966	1,84952
12:30-12:45	3,53273	-2,9939	3,67499	-2,4527	0,54219	-3,4678	-2,4198	3,0852	-3,8318	-2,0495	-1,9547	1,36616
12:45-13:00	1,31773	0,27251	-2,0885	1,70119	2,30735	-0,0576	3,85891	-0,31	-1,7998	-0,9074	-0,4884	-2,9547
13:00-13:15	-0,7971	-0,0221	1,32388	0,48622	-0,1361	-3,7127	-1,8443	-3,2359	-1,516	1,89812	-2,6523	1,77089
13:15-13:30	-0,2039	0,94617	1,03151	0,60646	-3,4793	3,22224	2,95363	2,55242	-1,6069	2,39437	0,59925	-0,6574
13:30-13:45	-2,0468	-0,3039	0,15488	-2,5674	-1,4693	-3,2437	3,2494	-3,251	-2,4366	1,29574	-2,4561	-1,1809
13:45-14:00	0,36496	2,19909	-1,5719	-1,4576	-3,1082	-1,3336	1,50011	-1,5458	0,47125	-2,5788	1,59537	1,60556
14:00-14:15	-0,0576	0,22947	-1,974	-1,2077	-1,0366	-3,0943	-1,3142	-2,849	-2,8987	-2,4924	-1,8443	-3,4176
14:15-14:30	3,79484	2,25013	-3,3988	2,63891	-2,7037	3,01502	2,90356	3,90476	2,25135	0,80902	-3,2722	0,64323
14:30-14:45	2,60247	0,71703	-0,9326	0,46926	-3,4738	2,69875	-0,2192	2,99318	1,89253	-0,8503	1,67662	1,94052
14:45-15:00	-2,3923	0,14846	3,27133	0,92009	2,8873	1,24371	1,42916	-1,2615	2,19725	1,27176	-1,2297	-3,1113
15:00-15:15	2,69775	-0,535	-0,8831	-0,5779	-3,5198	3,47604	-0,8312	-2,6103	-2,6282	1,05137	-0,9813	-3,6252
15:15-15:30	1,73684	0,16669	-3,3223	-0,2376	1,28662	1,64018	0,07832	-0,512	-0,6214	-3,0237	-1,6777	-2,4228
15:30-15:45	-1,6453	2,5392	3,44695	-1,6165	-2,7602	2,66215	-3,7286	-0,1298	-1,5729	3,35685	0,28238	-2,9818
15:45-16:00	0,93753	-1,6858	0,5159	2,00262	-2,1714	2,89616	-1,192	3,63051	2,03851	1,07347	1,90788	-1,5485
16:00-16:15	-0,4583	1,96924	-1,7791	2,72511	-3,2153	-0,3155	-0,3499	1,6055	2,32908	0,07045	1,099	-2,9019
16:15-16:30	-3,1384	-0,3041	-3,0465	1,87028	-2,951	1,03283	-0,2858	0,8571	-3,6443	2,06773	-2,1443	-3,0501
16:30-16:45	3,51964	3,23434	0,68318	3,00398	3,68208	-2,5185	2,49571	1,89448	1,35589	3,29619	1,77278	-1,0206
16:45-17:00	2,67244	-2,7797	2,04248	-0,1033	2,78743	1,10815	-1,7257	0,50513	-2,6813	2,20595	-0,6115	-2,1451
17:00-17:15	-2,9791	-2,9807	2,22449	1,00536	2,83178	0,52157	2,5086	-0,6231	3,11713	2,22209	0,21271	0,34648
17:15-17:30	2,47645	-0,7435	0,91239	-3,4081	1,98145	-1,2801	0,8497	1,85999	1,79796	-2,7761	-3,0629	-3,1517
17:30-17:45	0,55678	0,55143	-2,3579	-2,8147	-0,5646	-0,2781	3,58593	0,79865	2,24052	-2,8664	2,7019	0,06772
17:45-18:00	0,75546	-1,5062	2,73628	3,01852	-0,1457	1,69272	-0,9481	1,19171	0,34585	-0,681	2,70063	-2,3728
18:00-18:15	-1,6608	-1,1624	-1,083	0,00178	1,48993	0,81105	-3,1706	2,75904	-3,0233	0,64986	-1,2521	-2,9186
18:15-18:30	-2,2215	-1,5956	-1,3261	-3,3709	3,54791	-3,0634	-0,852	0,63822	1,66201	2,10309	1,68068	-0,7752
18:30-18:45	2,0053	-1,4139	1,27457	-3,3306	1,44281	1,04861	-1,1619	-0,7407	-2,4294	-2,3748	1,99226	-0,2
18:45-19:00	3,67697	-2,6222	0,64785	-1,5941	-2,2601	-1,2044	1,82469	1,08271	-0,4339	-2,1515	-2,2958	-0,681
19:00-19:15	-3,4539	-1,5989	0,90045	0,40484	2,36687	2,74208	3,65208	1,44041	0,77966	0,43922	0,24726	-2,2979
19:15-19:30	2,40415	1,80945	-0,1549	0,35857	-0,9555	-0,7355	1,09686	2,07995	0,08551	-1,3057	1,7399	-3,1883
19:30-19:45	-3,1231	0,38917	-0,3434	-0,9395	-1,0826	2,52034	-1,7688	-2,8321	1,09346	0,72517	-0,1083	-3,385
19:45-20:00	0,9089	-2,7543	1,81496	-0,0917	-0,579	-0,4333	6,6E+25	2,27573	0,83055	-2,202	1,45036	-2,0689
20:00-20:15	-2,0116	-2,0983	0,09865	-1,3176	1,96871	-2,7172	-2,0839	2,64323	1,49492	1,13419	2,7263	-1,5053
20:15-20:30	0,81427	-2,791	-1,3398	-2,7231	0,86702	1,28262	-2,5463	3,13448	3,03887	-1,3156	-1,305	-0,0018
20:30-20:45	-0,5768	-1,7803	-1,1838	1,43668	1,79294	3,16506	0,69997	0,8182	-0,7343	3,79667	2,0874	-3,1042
20:45-21:00	-1,3109	-1,6044	2,49488	2,33555	2,85945	1,20457	2,70393	0,08493	0,00472	-3,9078	-1,1001	-2,3482
21:00-21:15	2,63833	3,60764	-2,2607	-1,9443	1,18858	-2,8429	2,60307	1,32716	-0,4105	2,92975	-2,1538	0,86981
21:15-21:30	-2,7424	-0,543	-1,5635	-0,281	-0,3075	-0,4657	2,86605	-1,6041	-0,2841	-2,4026	0,91168	3,64788
21:30-21:45	2,21054	-1,9654	0,77043	-1,5396	-1,0445	-1,0849	-1,6722	0,23332	2,52502	-2,6672	1,16654	-1,2486
21:45-22:00	1,98067	-1,3873	0,32972	0,5494	2,62911	1,71895	2,28846	2,4453	-0,5171	-1,805	2,62057	-1,6338
22:00-22:15	-1,4042	-0,046	3,43409	-0,3045	2,32401	2,29798	3,89273	-0,3728	1,97589	2,03087	-1,127	2,18597
22:15-22:30	2,53211	-2,5075	-3,3673	-1,1481	2,29897	-2,6803	0,10744	3,67657	-1,4267	-3,2542	-1,1954	-3,558
22:30-22:45	-2,3644	-1,6512	-1,8368	2,00323	2,2243	3,5672	1,88836	3,47665	-2,8542	2,28815	-0,9514	-2,0961
22:45-23:00	-0,7098	0,49076	2,82775	3,34365	-2,9795	-1,1982	-1,8056	-1,4089	2,17129	1,35846	2,17867	-2,4252
23:00-23:15	3,41756	-2,2772	0,90637	2,31998	1,15133	3,23384	-0,4693	1,30949	-1,086	2,55633	3,17455	-0,4259
23:15-23:30	-1,1342	0,47963	-1E+24	-0,4172	-2,3002	-2,7669	1,91826	-0,7812	-3,2185	0,17102	-2,0601	-3,8408
23:30-23:45	-3,393	0,00722	-1,4717	0,65001	0,19903	1,18482	-0,8835	-3,0958	-2,9182	0,61151	0,16968	-1,5247
23:45-24:00	-0,9738	3,22974	1,30784	-0,4196	1,0707	0,3744	0,22069	1,98378	-0,9245	-3,7307	0,7398	0,26503

Table G.8: Weight matrix for quarter-hour embedding layer in speed module. All values multiplied by 10^{34} for readability. (cont.)