Demand-Responsive Transport to ensure accessibility, availability and reliability of rural public transport

OPEN-SOURCE BASED DATA WAREHOUSE

Output I2.1
Aalborg University
Department of Computer Science

Authors
Kristian Torp
Magnus N. Hansen

31/12/2021
Project nr. #R101
(Interreg Baltic Sea Region)

Table of Content

1 Overview

This report first discusses how General Transit Feed Specification (GTFS) [1] feeds from multiple countries or regions can be modeled, integrated, validated, and queried uniformly in a data warehouse. This includes the design of a generic data warehouse for storing GTFS feeds. GTFS data is available for most if not all EU countries.

The report then moves to describe how GNSS (or GPS) data from vehicles can be used to derive knowledge. The report focuses on two important Key Performance Indicators (KPIs) travel time and fuel consumption. These KPIs are the backbone of transport planning and evaluation in both public and private transportation organizations.

2 GTFS (General Transit Feed Specification)

This section describes the overall data warehouse design where GTFS data is stored. The data warehouse contains all information related to public transportation where the data is provided in the de facto standard GTFS format. This includes information on topics such as agency, stop, route, trips, stop times, fare attributes, fare rules, frequencies, and transfers. The data warehouse stores all the information that is provided by the various public transportation authorities. The data warehouse design supports multiple countries.

Please note that there are no GDPR issues related to using GTFS data. The data is already in the public domain and contains no information related to individual persons.

2.1 Platform

The PostgreSQL [2] relational database platform is used as the underlying relational database management system. This platform is chosen because it is widely used both by public and private organizations. The PostgreSQL platform is considered highly mature and is used in production environments in many organizations. Further, the PostgreSQL platform supports the Open Geospatial Consortium (OGC) [3] standard for handling spatial data via the PostGIS [4] extension. Finally, routing queries can be supported using the pgRouting [5] extension. The PostgreSQL platform and the two extensions PostGIS and pgRouting are all open-source products that freely can be used.

2.2 Data Sources

During the project, several different GTFS sources have been used. This covers three countries in the Denmark, Estonia, and Sweden.

2.3 GTFS Data Design

This section provides the details of the tables that are used for storing the. The overall design is shown in Figure 1. The scripts for creating the database are available online. In the following, the database design is described in further detail. GFTS covers buses, trains, and trams. For simplicity, we use the word bus in the following.

2.3.1 Table Description

In this section, we shortly describe each of the tables shown in the diagram in Figure 1.

- The *feed* table is the main entrance table. It contains the ID and geography that defines the spatial area covered by the data.
- The table *translations* contain language information.
- The table *shapes* describes the geometry of the various parts of a station
- The table *levels* contain information on various levels, e.g., if you need to go up and down.
- The table *agency* contains information on the agency.
- The table *fare_rules* describes the rules associated with a fare.
- The table *fare_attributes* describes the details for the fair_rules table.
- The table *payment_methods* is associated with fare_rules and describes how users can pay for the fare.
- The table *routes* describe the main routes, e.g. local or regional routes, or more specific as ferry or aerial lift.
- The table *routes_types* describes which types of routes are available.
- The table *trips* contain information on the trips on a route.
- The table *direction* describes the direction of a route. Typically there are two directions but there are also route variations.
- The table *frequencies* describe when the bus departures and arrives at the various stops.
- The table *calendar* adds specific date and time information to the bus.
- The table *calendar_types* contains various additional information on calenders, e.g., special country-specific holidays.
- The table *stop_times* is associated with when a specific bus stops are a specific stop at a specific route.
- The table *pickup_dropoff_type* contains information on the stops, e.g., schedule, phone assigned, or driver assigned.
- The table *stops* contain information on the individual stops.
- The table *location_type* contains information on the stops, e.g., boarding area, stop, platform, or generic node.
- The table *wheelchair_boarding* contains information on if a stop is accessible for wheelchairs users.
- The table *transfers* describe how a user can transfer to other routes at a specific stop.
- The table *transfer_type* contains details on how users can transfer at a stop.
- The table *attributes* contain details on a stop.
- The table *pathways* describe how to transfer when at a stop.

To make it simpler to learn to use the database, the names used in the GTFS specification have a far as possible been retained in the table names. However, reserved words in the SQL language are not used as table names. Similarly, are all column names taken directly from the specification.

Please note that the SQL scripts that create the data warehouse structures are a good source of information. These scripts are freely available for download.

2.3.2 Database Constraints

The database schema is implemented with as many database constraints as possible to avoid so-called dirty data. The general rules for adding database constraints are explained the following.

All tables have a primary key. This is listed in the GTFS specification and the implementation is aligned with the specification. Note that composite primary keys are used in several places.

Unique keys are added where the specification lists that values should be unique and a table already has a primary key.

Not-null contains are added where listed in the specification. However, the not-null constraint has been lifted on several columns because the data was not available in all GTFS feeds. To support most feeds the database implementation is not as strict as listed in the specifications.

Foreign keys are used in all places where they can be deduced from the specification. Foreign keys are very important to ensure that the data loaded is a coherent whole.

Foreign keys are not specified directly in the GTFS specification and the foreign keys used have been tested by loading feeds from three different countries. To the best of our knowledge, the implementation contains all the foreign keys possible. If the implementation contains too many foreign keys, these can be disabled individually. It is a simple technical task to disable a foreign-key constraint.

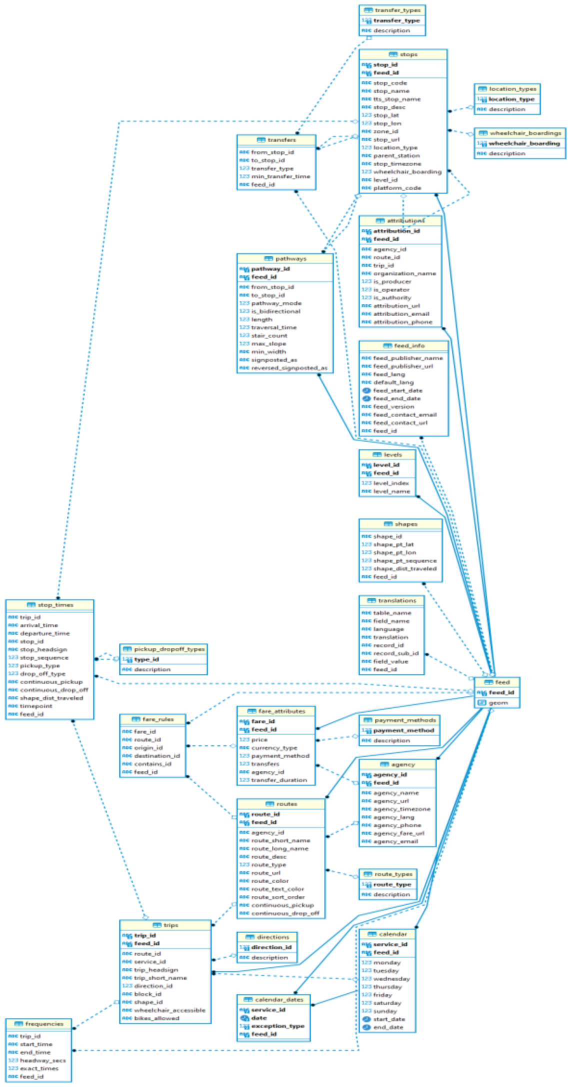


Figure 1 Overview of the GTFS Database

2.4 A load of a GTFS Feed

The database design described in Section 2.3 is loaded using a Python [6] script. The database design is non-trivial as it consists of a significant number of tables. The load-script is therefore also non-trivial. A particular problem is to ensure that the database constraints described in Section 2.3.2 are fulfilled.

2.4.1 Load Script Features

The GTFS feeds typically change weekly, monthly, or quarterly. This means that the content of the database quickly becomes outdated if the content cannot be updated simply. A complication of the load script is that that GTFS feeds can be incorrect. This has been experienced in several cases that an GTFS does not completely follow the GTFS specification.

The load-script accepts the following options/parameters.

- Create, this option creates the entire database shown in Figure 1. No data is loaded.
- Crop, this option drops the entire database, all data is naturally lost if the database is dropped.
- url, this option is a path on a local computer or a URL on the web address, the script with then load the GTFS feed specified by the URL.
- Reset, this option will first run the drop option and next create option.
- Validate, this validates a GTFS feed. To ensure that the GTFS data can be loaded into the database.
- Multi, this will load multiple GTFS feeds into the database.

Note that multi-option is needed because several countries have more than one GTFS feed for the country. The multi-option can also be used to load GTFS feeds from several countries into the same database.

2.4.2 Multi-Country Support

The database supports multiple countries or regions in several ways.

- The multi-option in the load script can be used. This will support multiple countries in the same database.
- The database connection can be changed. This will support for example one country or region per database.

In principle, the GTFS feeds for all EU countries can be loaded into the same data warehouse. However, to make it simpler to handle the data it is recommended to have a data warehouse for each country or each region for larger countries/cities.

The multi-country support also allows a region to load multiple versions of a GTFS feed into the same data warehouse. This makes it possible, e..g, to compare, a holiday season with summer or a winter season.

2.4.3 Testing of GTFS Feeds

The database design and the load scripts have been tested in the following way

- There are Python unit-test scripts that check for the typical errors.
- The database has been populated with data from three countries
- The SQL script has been adopted to accommodate typical errors in the GTFS feeds.

3 GPS Data Warehouse

GPS data is available from several sources. However, due to General Data Protection Regulation (GDPR) rules in the EU, there are restrictions related to acquiring, storing, and displaying GPS data. To comply fully with the GDPR rules we only use data in the RESPONSE project where we are certain that we are allowed to use it. However, the solutions provided in the project can be used in any country where there is a digital map from OpenStreetMap (OSM) [7] and where a transport organization has GPS data that they are allowed to use internally. In practice, OSM is available for the entire EU region.

3.1 GPS Data Design

The GPS data warehouse is based on an existing design shown in Figure 2. In the RESPONSE project, we built on top of this data warehouse design [8]. The contributions made during the RESPONSE project are the following and related to KPIs travel time and fuel estimation.

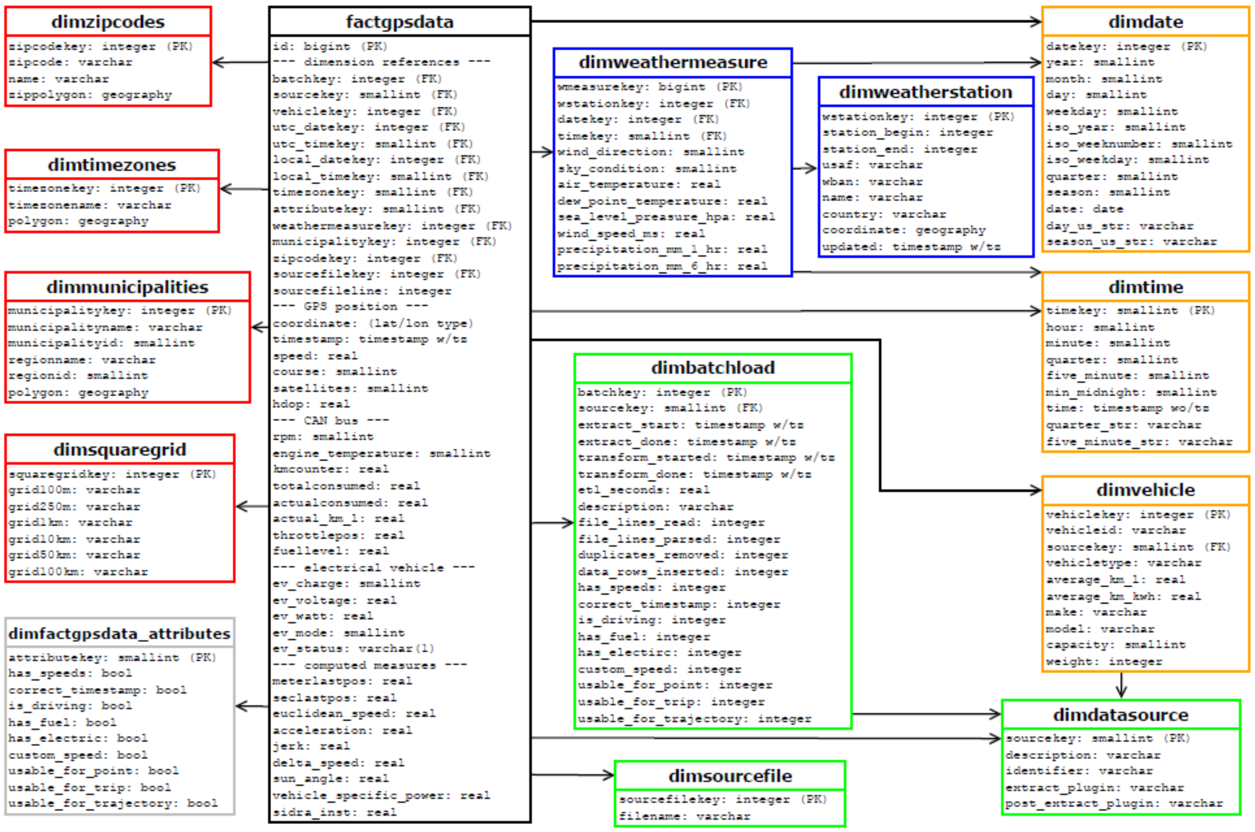


Figure 2 Overview of GPS Database

- Five different models for computing travel time. The models are machine-learning-based methods that enable travel time to be computed in areas with limited GPS data. The models are gradually more and more sophisticated. These models are fully described and documented in Appendix A
- Two different approaches for computing fuel estimations from GPS data.
- An implementation of a RESTful API on top of the extended data warehouse to query data programmatically, from basically any programming language. A complete set of examples are provided in the popular Python programming language. This API and examples are explained in detail in output O2.4 from the RESPONSE project.

- A web front-end for visualization of four important queries on the data warehouse (a) routing, (b) isochrones (3) travel-time cost matrix, and (4) traveling salesperson. The web-front-end and the four queries are explained in detail output O3.1 from the RESPONSE project.

3.2 GPS Data Used

To ensure compliance with the GDPR rules only existing data from Denmark are stored in the data warehouse. However, the data warehouse is generic and can be used to store GPS data from any country where OSM provides a digital road map. This is most of the World including the entire EU region.
The GPS trajectories are mapped to OSM road segments and the travel time is inferred from the mappings.

3.3 Travel-Time Estimation

Both estimation types of travel-time estimation models use travel times on OSM road segments.
The baseline model requires the average speed-limit delta for each road category. For instance, if we assume the speed limit on motorways in Denmark is 130km/h, and the average speed on motorways is 125km/h, the motorway speed limit delta is -5km/h. The travel-time estimation for motorway segments can therefore be calculated as $\frac{length}{speedlimit+(-5)}$. This model is simple, however, it generates good baseline results.

For the other models we use a deep-learning approach to train the models, the models are configured with ten layers of 100 neurons each and trained for about 8 hours each. The simplest deep-learning model is trained with the same set of features as the baseline model, that is length, speed limit (km/h), and road category. In (M1-M4) and the Baseline Model

Table 2 the performances of each model are displayed in different metrics, in each case lower is better (See Appendix A). There is a significant improvement in every metric from the Baseline model to M1. Model M2 adds all road-segment metadata (tortuosity and one-way). Model M3 adds road annotation data (crossing, calming). Model M4 adds temporal features (month, weekday, hour of day). A common trait of deep-learning models is that they require large amounts of training data. The results in (M1-M4) and the Baseline Model

Table 2 are for models trained with 45.5 million samples collected in Denmark from 2012-2014. An analysis of the accuracies are at different sample sizes is shown in Figure 1. Models with feature set D only outperform other models when trained on the full data set. From the line plots, we can expect only feature set D will improve with more data, but also that a 10-fold amount of data increase is required to gain a one percent improvement in MAPE. Since it is not trivial to collect these amounts of data, and the potential improvement is small, a better option could be to try models which are less dependent on large amounts of training data and add more features, such as grade of the road, weather, and details about adjacent road segments.

To use the models through the RESTful API and by extension the web front-end, the travel times of each segment in the road network are precomputed and stored in the database. From the database, the different routing functions can efficiently run using pgRouting. Model M4, however, requires a travel time for each combination of temporal features i.e. month, weekday, and hour. Therefore storing the model requires 2016 times more space. Using model D in the RESTful API is, therefore, slower than other models.

The models were applied to the road networks of Denmark, the Faroe Islands, and Sweden.

Model	Set	Length	Kph	Category	Tortuosity	One-way	Calming	Crossing	Month	Weekday	Hour
Baseline	A	✓	✓	✓							
M1	A	✓	✓	✓							
M2	B	✓	✓	✓	✓	✓					
M3	C	✓	✓	✓	✓	✓	✓	✓			
M4	D	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 1 Features used in Each Model (M1-M4) and the Baseline Model

Model	MSE	MAE	MAPE
Baseline	499.537	4.422	35.051
M1	46.317	3.128	27.903
M2	40.982	2.897	25.585
M3	40.922	2.884	25.339
M4	40.047	2.864	25.289

Table 2 Models Performance on the Test Set

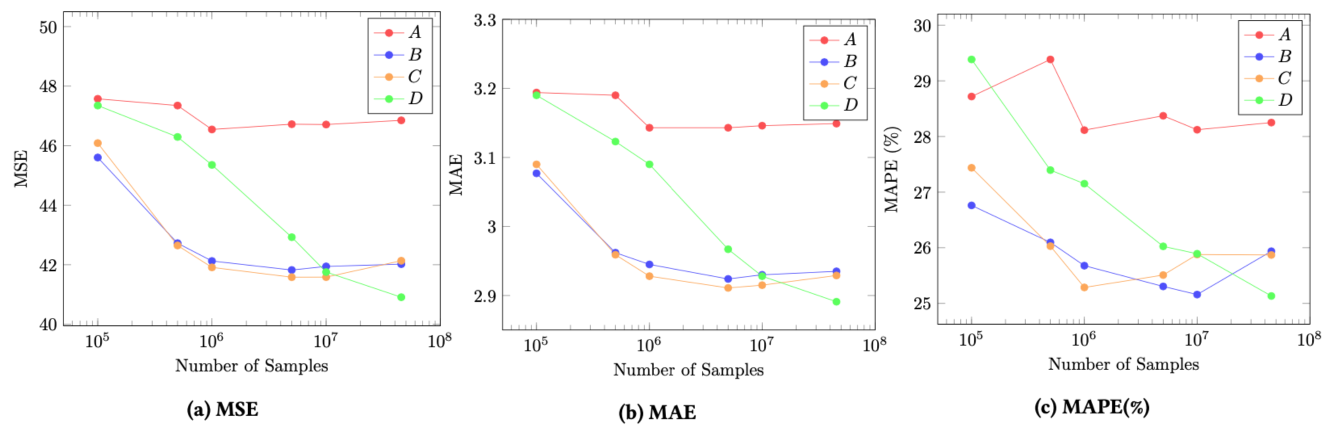
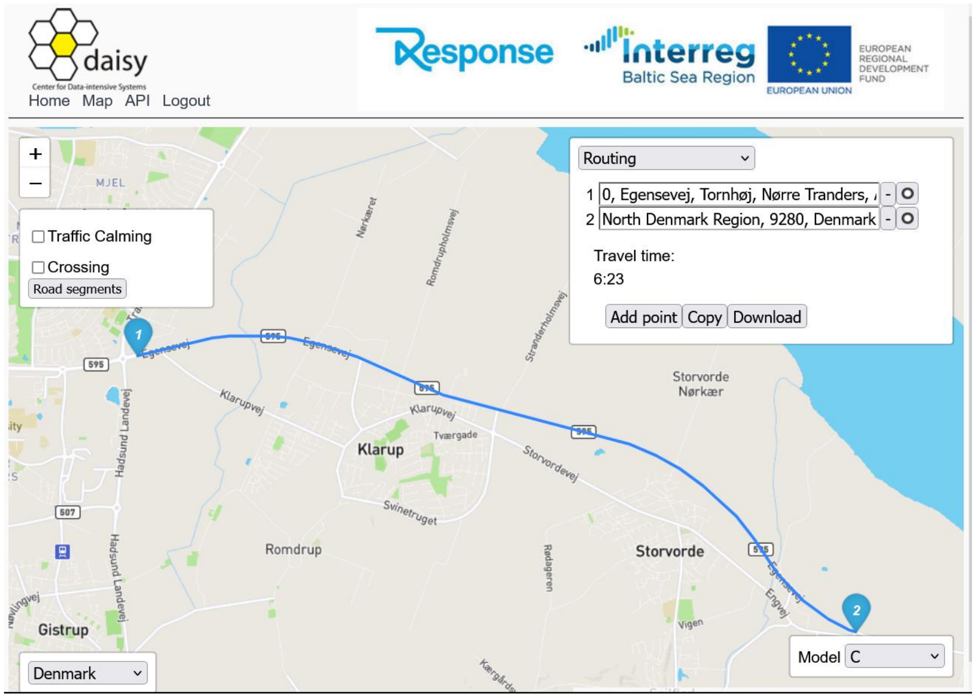


Figure 3 Metrics Over Sample Size

3.4 Fuel Consumption Estimation

During the RESPONSE project, the UN SDG goals have become increasingly more important being able to estimate the fuel consumption, e.g., on a route is a highly interesting way to use the data from PTA and enable political decision-making to consider different options and to identify market potential for private transport providers.

It is possible to collect fuel consumption data from the vehicles. However, this is expensive as additional equipment/hardware must be installed in each vehicle. Fortunately, a wide range of fuel-estimation tools exists. These tools can take high-frequent GPS data and convert this data to a fuel estimate. As an example, Figure 4 shows a route in a rural area using the web tool that will be introduced in Output O3.1. For this route, the estimated fuel consumption is shown in Table 3.



Routing

1 0, Egensevej, Tornhøj, Nørre Tranders,

2 North Denmark Region, 9280, Denmark

Travel time:
6:23

Model C

Denmark

Figure 4 Finding the Fuel Estimate on Rural Route Shown usring RESPONSE Web Front End

Table 3 Fuel Estimates for Rural Route

Length (m)	Trips	Sidra Estimate (l)	SP Estimate
8,720	424	0.98	2988.21

Table 3 shows that the route is 8,720 meters. That 424 trips have traveled this route on weekdays and that the estimated fuel consumption using the Sidra model [9] is 0.98 liters. Similarly, the fuel estimate using the SP model [10] is 2988.21 (no units on this type estimate).

To estimate the fuel consumption the following GPS data is required

- A vehicle ID
- A position (latitude, longitude)
- A timestamp to seconds granularity
- An estimated weight of the vehicle
- The slope of the road segments (optional)

A major legal challenge using GPS data for estimating fuel is the GDPR rules. To avoid all issues related to GDPR we only demonstrated the fuel estimation approach for Denmark as high-frequent GPS data is here available for research purposes. All techniques used in the RESPONSE project can be used anywhere GPS data and an OSM digital map is available. The OSM map is available for the entire EU region.

3.5 Role-Based Access to GPS Data

As mentioned in Section 3 access to GPS data must be restricted to comply with the GDPR rules. To comply with these rules the role-based access to the GPS data has been moved from the SQL level (the database level) to the RESTful API level (the application level).

The effect of moving from the SQL to the API level is that it can be ensured in programming code that no users get access to detailed information on a single or small set of drivers. The RESTful implementation will not return a result if the result is based on data from 10 or fewer vehicles.

There are currently two roles used at the API level

- A *public role* that can access the four major types of functionality routing, isochrones cost-matrix, and traveling salesperson. This role is given to the public meaning that all users that go to the website https://mapapi.cs.aau.dk/ can use this functionality without any login. The functionality is available both from a graphical user interface and via the RESTful API.
- An *API role* that can access the full RESTful API. Here the user needs to be registered to log in and gain access to the full API. However, all users will only get access to data that is aggregated from 11 or more vehicles.

By design, the public role can be used directly by public authorities and political decision-makers. The API role is targeted towards public and private mobility service providers with knowledge of querying map data.

A clear benefit of user access at the RESTful API level is that no user can overload the data warehouse structures as the number of requests a single user can issue can be restricted. Currently, any user can only make four requests per second. This number has been found by examining the response time of the most expensive queries.

4 Conclusion

For the GTFS data the project supports all data provided by a GTFS feed, e.g., agency, stop, route, trips, stop times, fare attributes, fare rules, frequencies, and transfers. The GTFS database can be set up by individual regions or hosted by a cloud vendor. The GTFS database contains no person-specific information and can be made freely available on the internet as there are no GDPR issues. The GTFS data contains the latitude and longitude of stops and routes and can thus easily be displayed on a map.

The source code for the ETL programs and the GTFS database is available online for download such that these programs can be maintained by the regions after the RESPONSE project finishes. The ETL programs are written in the open-source Python3 programming language and documented at a function and class level. Further, how to get started with loading a GTFS feed is provided.

For GPS data the GDPR rules apply and it has been a major issue to provide access to the two KPIs travel time and fuel-consumption estimates. For GPDR reasons the GPS data warehouse contains no information on drivers or vehicles. The travel-time estimation works for all of Denmark, all of Sweden, and all of the Faroe Islands. To avoid any GDPR issues the fuel estimation techniques are available for Denmark. However, both the travel time and fuel estimation approaches presented in the report can be used anywhere, e.g., the entire EU region. The technical details of using GPS data is provided in Appendix A in the scientific format typical for the research area.

The GPS data can be queried using a web-based graphical user interface or a RESTful API. The RESTful API is introduced in detail in Output O2.4 from the RESPONSE project. The web-based graphical user interface is described in Output 3.1 from the RESPONSE project.

5 References

[1] GTFS, "GTFS: Making Public Transit Data Universally Accessible," [Online]. Available: https://gtfs.org/.

[2] PostgreSQL, "PostgreSQL Homepage," [Online]. Available: https://www.postgresql.org/.

[3] OGC, "Open Geospatial Consortiuim," [Online]. Available: https://www.ogc.org/.

[4] PostGIS, "PostGIS," [Online]. Available: http://postgis.net/.

[5] pgRouting, "pgRouting," [Online]. Available: https://pgrouting.org/.

[6] Python, "Python Homepage," [Online]. Available: https://www.python.org/.

[7] OpenStreetMap, "OpenStreetMap," [Online]. Available: https://www.openstreetmap.org/.

[8] O. Andersen, B. K. Benjamin, T. Christian and T. Kristian, "An Advanced Data Warehouse for Integrating Large Sets of GPS Data," in *DOLAP*, 2014.

[9] A. R. B. D. A. R. R. B. Bowyer D, "Guide to fuel consumption analyses for urban traffic managemetn," Australian Road Research Board, 1985.

[10] J.-P. J. L., "Understanding and quantifying motor vehicle emissions with vehicle," PhD thesis, Massachusetts Institute of Technology, 1998.

Appendix A

Travel-Time Estimation in Rural Areas

Magnus N. Hansen
Aalborg University
Dept. of Computer Science
mnha@cs.aau.dk

Kristian Torp
Aalborg University
Dept. of Computer Science
torp@cs.aau.dk

ABSTRACT

The travel time between a source and a destination in a road network is a very important metric in many planning applications. Several commercial solutions exist and the use of these solutions in city areas with congestion problems is important for both private and public transport companies. However, these companies are also worried about the price of using commercial solutions in rural areas or during non-rush hours, because travel time varies very little. In this paper, we build four deep-learning models for predicting travel time in rural areas. The models are trained using incrementally larger feature sets. The simplest model uses length, road category, and speed limit. The most advanced model uses multiple features from the road network such as speed bumps and pedestrian crossings. All models are trained using GPS data in a privacy (GDPR) compliant format. Intensive experimentation with different GPS dataset sizes is conducted to determine how this influences the accuracy of the models. All four models are trained using the full road network of Denmark. We show that travel time can be estimated accurately in rural areas 25.3% MAPE and that even a few road-network annotations increase the accuracy. The results show that even small GPS data sizes contribute considerably to increased accuracy and that a simple baseline can be accurate on rural routes.

KEYWORDS

travel time, map data, GPS

ACM Reference Format:
Magnus N. Hansen and Kristian Torp. 2021. Travel-Time Estimation in Rural Areas. In *Proceedings of* . ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Travel time is a very important metric in many types of transport applications such as route planning and parcel delivery. A wide array of different sensor data is being collected, e.g., GPS, induction loop, and LIDAR radar. Such data can significantly improve the accuracy of travel-time estimations [2, 7, 9, 19]. Common to all the sensors is that there is a non-trivial up-front investment in hardware and software. An alternative is to use online commercial travel-time services from providers such as Google, HERE, or TomTom. However, using such services naturally comes at a monetary cost.

Public administration at the state, regions, and municipality level that do transport planning are very aware of the cost-benefit of using the commercial travel-time services. In particular, in rural areas with limited congestion problems the benefit of using the commercial solutions is being questioned. Furthermore, as you typically pay per request both public and private transport companies are also looking for alternatives.

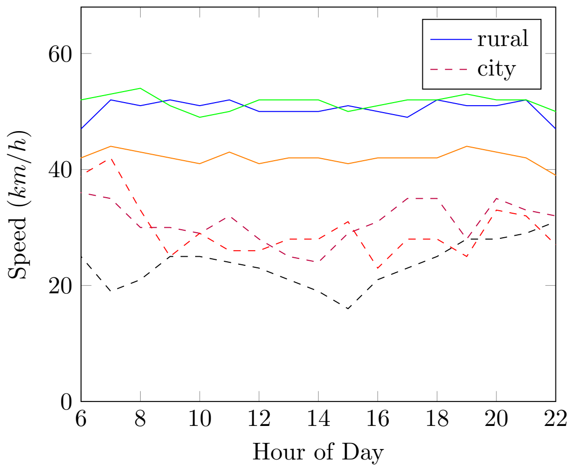


Figure 1: Travel Time in Rural and City Areas, 50 km/h

In rural areas, the speed on a road segment (and therefore the travel time) is often very stable. Figure 1 shows the hourly speed from road segments in city and rural areas of Denmark between 6:00 and 22:00. There are between 3 and 297 trips per hour on each segment collected over 24 months. The speed limit is 50 km/h the dashed lines are in cities and the solid lines are in rural areas. During the day the speed is stable, there are minor fluctuations during the evening. The results in Figure 1 indicate that the travel time on segments can be predicted accurately in rural areas and these segment-level predictions can be used in routing algorithms to make accurate travel-time predictions.

A negative consequence of it being very easy to collect very large GPS datasets is that the privacy of drivers can be violated. In the EU-zone, the GDPR rights put strict limits on which information about drivers can be collected and shared. To be GDPR compliant, we use a data foundation that is as fine-grained as possible but makes it very hard (impossible) to track individual drivers as it is possible if GPS trajectories are used [12, 20]

The contributions of the paper are as follows.

- We show how to accurately estimate travel time in rural areas based only on the road category, the speed limit, and the segment length.
- We show how the travel time estimated can be improved by adding information such as speed bumps and pedestrian crossings.
- We show how larger and larger GPS datasets can improve the accuracy of the travel-time estimation.
- We implement the four proposed models in three countries, Denmark, Faroe Islands, and Sweden.

The remaining parts of the paper are organized as follows. section 2 looks at related work. The data foundation for the four models is presented in section 3. The machine-learning models and how they are trained is discussed in section 4. The results are presented in section 5. section 6 concludes the paper and points to directions for future research.

2 RELATED WORK

In [13] they use trajectories and secondary attributes such as vehicle type, day-of-week, time-of-day, and rainfall. Their trajectories are split into sub-paths. They do not use road attributes. They use both taxi and bus data. They use a neural model and their Attribute Correlation Module finds the correlation between the attributes and the travel time.

In [16, 21] they use tensor factorization to estimate travel times. Each travel-time tensor has three dimensions: time slot, road segment, and driver. Each road segment has features such as the number of lanes, length, and points of interest such as schools, offices, shopping near the segment. Each time slot has information about the number of vehicles traversing an area of road segments.

In [17] they use the same taxi dataset and road features as in [16] and [21], but use a deep neural network architecture instead of tensor factorization. Their sparse denoising auto-encoder is designed to estimate travel time without depending on a large labeled training dataset. By intentionally adding noise in the training data, they create a model that is robust to erroneous and missing features. Their input training vectors contain features about the each road segment but also the features of adjacent road segments. This allows them to model the dependencies of adjacent and connected road segments.

In [23] they use both historical and sparse real-time trajectory data, which they map onto road segments or pathlets. A pathlet is a sequence of concatenated road segments, which replace the road segment edges, reducing the overall number of edges in the road network graph. They use matrix factorization [5] to create a latent representation of road segments and pathlets, which is used to train a Gaussian Process [15] model that can estimate travel times.

Prediction of travel time assuming limited GPS data available is also the topic in [12, 18]. A major difference to [12] is that the focus is on urban travel time and there is special focus on congestion. Further, road network annotation such as speed bumps are not considered. Very large cities (New York and Shanghai) are also the focus in [18] and the variation in travel time (called *temporal dynamics* is a focus point.

Dynamic travel-time prediction using GPS data and machine learning is the topic in [1]. A complete system that computes dynamic travel times is presented. The system is tested in three large cities. In this paper we focus on rural areas and assume that the travel time does not need to be updated while driving. Dynamic travel time is also the topic in [22]. Major differences are that they assume very large dataset and used histogram for weight on road segments or sequences of road segments.

3 DATA FOUNDATION

The map data is a graph $G(V,E)$ where G is a graph consisting of V vertices and E edges. We assume that for each $e \in E$ $e = (eid, category, speed\text{-}limit, geom)$. Where eid is a unique edge ID, $category$ is the OSM category, $speed\text{-}limit$ is the speed limit, and $geom$ is the geometry in the form of a linestring. If OSM does not provide a $speed\text{-}limit$ a speed limit is derived from the $category$ column.

The map annotation data $m \in M$ $m = (pos, type, subtype)$, where pos is a position, i.e., a (lat, lon) pair and $type$ is an annotation type, e.g., speed bump or pedestrian crossing. This information is extracted from OSM and mapped to the nearest edge e in the map using the position pos. This mapping results in the $m' = (eid, type, subtype)$ where the pos is replaced by an edge ID eid.

We examine speed bumps because they are designed to lower the driving speed and therefore influence the travel time. We examine pedestrian crossing because cars must yield to pedestrians, which also influences travel time. Note that the speed bump and crossing information is not complete on the OSM maps. An informal study af speed bumps showed nearly complete coverage in one Danish municipality and estimated less than 50% in another Danish municipality. Table 1 shows the number of road segments with speed-bump/crossing information and the number of rows with GPS data on these segments, see Table 2. A city zone is defined as a city with more than 250 inhabitants, otherwise, it is a land zone.

Annotation	Zone	Segments	Rows
speed bump	city	1348	183,139
	rural	242	71,497
crossing	city	430	55,269
	rural	28	841

Table 1: Speed Bump/Crossing Road Segments and Rows

The GPS data $d \in D$ $m = (eid, month\text{-}time, travel\text{-}time)$, where eid is the edge ID, $month\text{-}time$ is a rounded timestamp consisting of a month, e.g., December, a weekday, e.g., Tuesday, and an hour of the day, and $travel\text{-}time$ is the travel time on the edge in seconds. The data is aggregated from GPS data sampled with 1-second sampling period.

An example of the GPS data is shown in Table 2. The unique key for the table is the combination of the first four columns. The table shows $travel\text{-}time$ on Fridays in the month of April in the period 5:00 to 9:00 for the edge-id (eid) 444463. This edge is a 394.3-meter long road segment of the road category *secondary* in a rural area where the speed limit is 80 km/h. The road is broad and straight, which leads to speeds are an average of ~95 km/h. Please note that we do not consider the travel direction because we focus on rural areas and the travel time is assumed to be identical in both directions for road segments where two-way traffic is allowed.

eid	month	weekday	hour	travel time (s)
444463	April	Friday	05:00	15.00
444463	April	Friday	06:00	16.00
444463	April	Friday	07:00	17.48
444463	April	Friday	08:00	17.09
444463	April	Friday	09:00	17.67

Table 2: Example of GPS-Based Travel Time

The temporal information in Table 2 is rounded to ensure the privacy of the drivers. This means that we do not have trajectory data available, e.g., to follow one specific trip over multiple road segments (edges). The aggregation of the travel time at a road segment level makes it simple to combine multiple GPS data sources where the sampling period is different, e.g., 1, 2, 5, or 60 seconds.

We use GPS data from a field trial conducted during the years 2012-2014 a total of 471 vehicles participated. The distribution of observations per segment category is shown in Figure 2. Table 3 shows the number of rows (**#Rows**), number of segments (**#Seg**), the total length in kilometer (**Len**) the average number of rows per category (**Avg**), and the percentage of road segments (**%**). In this paper, we focus on the 12 main road categories in OSM, and shown in Figure 2. Only very small roads are excluded, e.g., service roads (cemeteries/parks) and dirt roads.

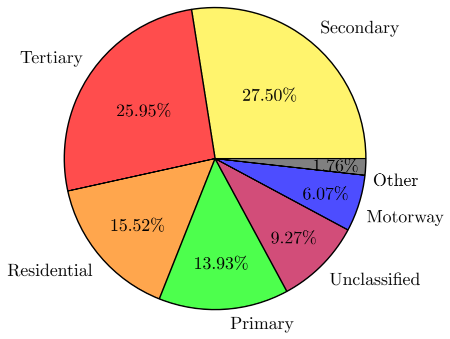


Figure 2: Distribution of GPS Data per Road Category

Category	#Rows	Len	#Seg	Avg	%
secondary	12.3M	4703	21,050	586	72.7
tertiary	11.6M	4088	24,866	468	44.2
residential	6.9M	3195	30,989	225	10.8
primary	6.3M	2535	11,220	557	85.7
unclassified	4.2M	6692	24,448	170	18.6
motorway	2.7M	2270	2,117	1287	95.7
motorway_link	462K	395	1,039	445	60.2
trunk	293K	412	774	378	88.2
secondary_link	18K	7	89	197	52.3
trunk_link	11K	31	103	106	44.0
primary_link	7K	18	104	71	45.6
tertiary_link	2K	3	31	66	46.2

Table 3: GPS Data per Road Category

It is an assumption that we only have GPS data on a small fraction of the road network, i.e., GPS data is a scarce resource.

To support the hypothesis that travel time is easier to estimate in rural areas than in city areas Figure 3 shows the distribution of coefficient-of-variation (CV) for different road categories in rural areas (red) and in cities (blue). For road segments of the *motorway* category, the CV is fairly similar. The same applies to the *trunk* category (not shown). However, for the *primary* and *secondary* categories shown in Figure 3, the CV is lower in rural areas. The same applies to the *tertiary* and *unclassified* categories (not shown). The *residential* category is only in city areas. For the various link categories there are too few road segments to make a clear conclusion.

4 MODELS

In this chapter, the data cleaning process is first presented. Next, it is explained how the features are encoded, such that they can be used for training neural network models. Finally, the training of the neural networks is discussed. This includes choices of hyperparameters and optimizations method.

4.1 Data Cleaning

To bring relevant information about each edge, the three data sets GPS, map, and map annotation are joined on the *eid* column to form a single data set. Then the following set of filters are applied in the order specified. The percentage of rows removed specified is relative to the total samples left from the previous step.

(1) **Categories**: The minor road categories are removed: *service*, *unpaved*, *living street*, *track*, *ferry*, and *road*. Removes 953,453 rows (1.62%).
(2) **Min observations**: Due to GDPR a minimum of seven observations per segment is required. Removes 214,911 rows (0.37%).
(3) **Gaussian Mixture**: Inspired by [17] we fit three multi-variable Gaussian to the logarithm of road length and travel time, and remove the 1% least likelihood data, as shown in Figure 4. Removes 577,325 rows (1%).
(4) **Max speed**: Max 60% above the speed limit. The length of each edge is extracted from its geometry, and together with the travel time, the average edge speed is calculated. Due to GPS and map-matching noise the edge speed is unreasonably high in some samples. Removes 1,054,793 rows (1.85%).

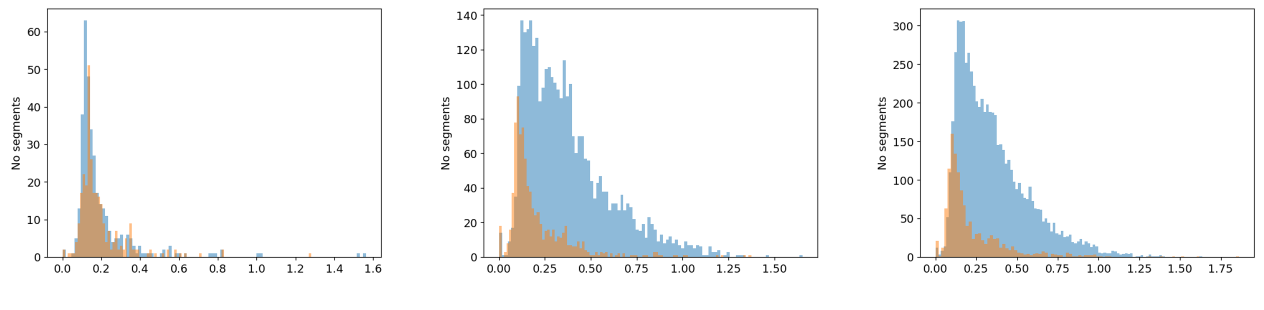
In total 2,800,482 rows of 58,900,953 rows are removed from the data set, corresponding to 4.75%.

4.2 Features and Their Encoding

The neural-network models are standard fully-connected networks, which require the input and output to have a fixed size [6]. The output is a travel-time prediction, which always is a single scalar value. The input consists of the features associated with an edge e in the graph G. The data originates from the three data sets described in section 3

In the map data, each edge contains a *category*, *speed-limit*, a *one-way* tag, and the linestring *geom*. The *speed-limit* is a scalar and the *one-way* tag is converted to a Boolean.

The *category* can either be labeled encoded or one-hot encoded. Label-encoding keeps the feature as a single scalar and can help the model derive an ordering of the categories. One-hot encoding transforms the categories to an input per category (in our case we have 12 categories). One-hot encoding suggests no ordering of the categories, and we did not find an obvious ordering of the categories. From experiments the higher input dimensionality of one-hot encoding is not an issue, therefore one-hot encoding is chosen.



(a) Motorway (b) Primary (c) Secondary

Figure 3: Distribution of Covariance per Road Category, Blue = City, Red = Rural

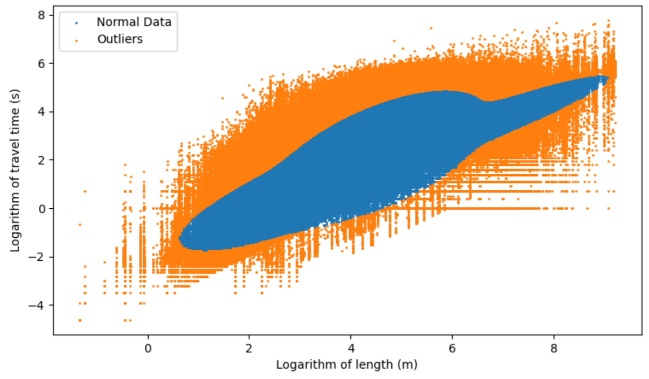


Figure 4: Anomaly detection with Gaussian Mixture Model

Feature	Source	Source Enc.	Enc.	Domain
Length	Map	Float	Float	1.1 - 9930.0
Speed limit	Map	Integer	Integer	10 - 130
Category	Map	String	One-hot	12 categories
Tortuosity	Map	Geometry	Float	1.0 - 44.06
One-way	Map	Boolean	Boolean	0 or 1
Calming	Ann.	Relation	Integer	0 - 2
Crossing	Ann.	Relation	Integer	0 - 2
Month	GPS	Datetime	Integer	1 - 12
Weekday	GPS	Datetime	Integer	1 - 7
Hour	GPS	Datetime	Integer	0 - 23

Table 4: Feature Encoding

The *geom* column is of different size because the linestrings consists of a varying number of points. The varying size makes the *geom* column difficult to use directly as an input to the model. However, the length and tortuosity are fixed size scalars and are extracted from the *geom*.

Other researchers [16, 17] use tortuosity for the same purpose of travel-time prediction. The tortuosity is a measure of how twisted a linestring is. It is defined as $tor = length/d$, where d is the straight-line distance between the start and end of the linestring and *length* is the actual length (using the *ST_Length* function). Figure 5 shows the most tortuous road segment in the map data set, with a tortuosity ratio of 44. Figure 6 show the distribution of tortuosities. Less than 0.1% of road segments have a tortuosity above two, hence these segments are not represented on the graph.

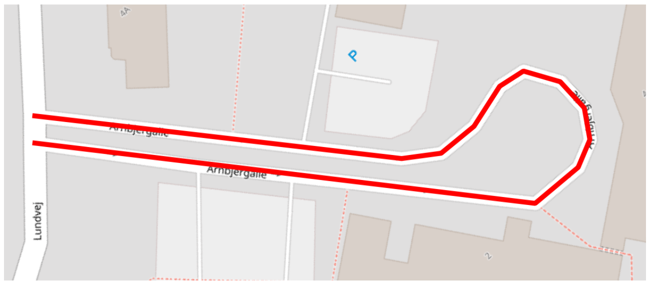


Figure 5: The Most Tortuous Segment in our Map Data Set

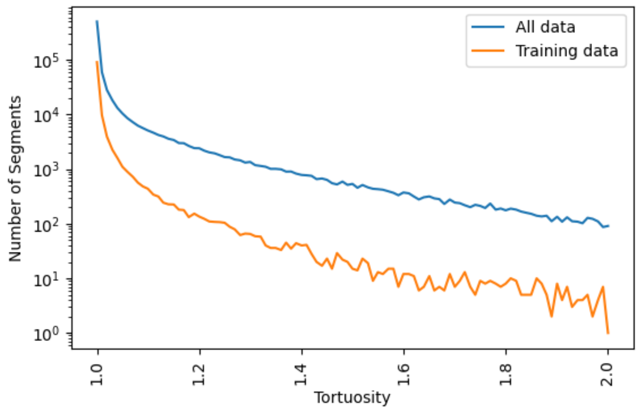


Figure 6: Distribution of Tortuosity

Each row m in the map annotation data M is mapped with *pos* to the nearest edge *eid* in the map G. For each edge e the number of annotations and their type is counted. This results in two integers for each edge; *calming* and *crossing*.

The GPS data D contains the $month - time$ which is encoded as three integers: *month*, *weekday*, and *hour*. These three temporal features are cyclical, therefore experiments are conducted with the features each transformed to a sine and cosine value on a unit circle. With this transformation, the distance relationship between, the values at the end and the start of a cycle are preserved. However, the experiments show lower accuracy with this transformation, which indicates the models can better learn the representation of cyclical features, as a single input instead of two for each of the *month*, *weekday*, and *hour* values. Finally, the travel time is extracted as a float.

4.3 Models and Training

The focus while developing the travel time model is to find the best model using existing and simple neural network architectures. Others have developed matrix-decomposition [11] and auto-encoder models with good results [16]. These papers use GPS data from the urban road network of Beijing, and our data set primarily covers rural Denmark. Due to the more complicated problem of predict congestions in urban road networks than the problem with travel time in rural areas as is the focus of this paper, good performance is possible even with a simple neural-network model architecture and well-tuned hyperparameters.

4.3.1 Baseline Model. A baseline is created to validate that the neural-network models can outperform the simplest model possible. We experiment with a baseline model that assumes vehicles always drive at exactly the speed limit. This model only requires the length and speed limit of each road segment. However, such a model is only usable on road categories such as motorways and trunks where constant speed near the limit is possible. To make the baseline model more realistic the *travel-time* is adjusted, by the mean deviation of the mean *driven-speed* and *speed-limit* for each road *category* computed from the GPS data. This is defined as follows.

$$e_i.speed = \frac{e_i.length}{e_i.travel\text{-}time}$$

$$cat_j.speed\text{-}delta = \frac{1}{|C_j|} \sum_{k \in C_j} e_k.speed - e_k.speed\text{-}limit$$

$$e_i.tt\text{-}estimate = \frac{e_i.length}{e_i.speed\text{-}limit + cat_j.speed\text{-}delta}$$

Where $e_i.travel\text{-}time$ is the average travel-time on edge e_i computed from the GPS data as discussed in section 3 and shown in Table 2. $e_i.length$ is the length of edge e_i and $e_i.speed$ is the (computed) average speed.

$cat_j.speed\text{-}delta$ is the average speed-limit difference of all edged where the *category* is j and the speed computed on edge e_i of the road category j of road segment of that *category*. The estimated travel time $e_i.tt\text{-}estimate$ is calculated from the *length* and *speed-limit* on edge e_i adjusted by the *category* mean speed-limit difference $cat_j.speed\text{-}delta$.

4.3.2 Neural-Network Models. The other models are multi-layered fully-connected neural networks. The input layer size ranges from 14 to 21, depending on the set of features, as shown in Table 5. The output layer is a single scalar which is the predicted travel-time.

The data is randomly split into a training, validation, and test set in the ratio 3:1:1 [17].

The features, including travel-time, are of varying domain sizes, as shown in Table 4, therefore to stabilize model training all features are min-max normalized individually [4] on the training set. All inputs and outputs are therefore in the range 0-1.

For evaluating the impact of the number of features, four feature set tiers are created, as shown in Table 5. Feature set A contains length, speed limit, and category where the 12 categories are one-hot encoded. This adds one extra input per category. Feature set B has all available road-segment metadata (length, speed limit, tortuosity, one-way, category). Feature set C adds road annotation data (crossing, calming). Finally, feature set D adds temporal features (month, weekday, hour).

From experiments with the L1 and L2 loss functions, the L1 loss has a slightly worse MSE about 7% compared to the L2 loss, however L1 loss improve MAE by 11% and MAPE by 45% compared to the L2 loss. Therefore, L1 loss is chosen.

ReLU [6] is used as the activation function. The optimizers SGD (Stochastic Gradient Descent) [6], Adam [8], and AdamW [14] are evaluated with experiments. AdamW is best at reducing the number of epochs required for the loss to converge, and is therefore used for training all models. The default β, ϵ and weight decay values of the AdamW implementation in PyTorch are used ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $weightdecay = 0.01$).

None of the models use dropout, but simple experiments with dropout applied to the hidden layers show, that models trained with dropout perform worse.

The hyper-parameters learning rate, number of layers, number of neurons per layers, and batch size are tuned with the Bayesian Optimization with the goal of minimizing the validation loss. The implementation of Weights and Biases [3] is used for this purpose. Weights and Biases is used extensively during development to keep track of the models. The best learning-rate is 0.001, but the other three hyperparameters all tend to perform best with large values. Low batch sizes result in the models converging quickly, but larger batch sizes tend to give better results after a longer training time.

We find ten layers, 100 neurons per layer, and batch size 70,000 can fit into our GPU RAM (11GB), and outperform other configurations. Our models are therefore bottle-necked by the size of GPU RAM. This drawback however allows experiments of models trained with other parameters, such as feature set and number training samples where larger models require more time to do the same experiments.

For each training run only the model state with the lowest validation loss is saved and used for evaluation.

Model	Length	Kph	Category	Tortuosity	One-way	Calming	Crossing	Month	Weekday	Hour
Baseline	✓	✓	✓							
M1	✓	✓	✓							
M2	✓	✓	✓	✓	✓					
M3	✓	✓	✓	✓	✓	✓	✓			
M4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 5: Models and Feature Sets

5 RESULTS

In this section, we present the results using the baseline model and the four neural-network models. We look at the accuracy of the results at an overall level, at a road category level, and at a route level, i.e., a number of road segment travel in sequence. The latter is of the highest interest to end users such as transport companies. Finally, we look at how the size of the GPS datasets available affects the accuracy.

5.1 Hardware

All experiments are conducted on the same physical hardware. The back-propagation neural network (BPNN) models in Table 6 each take approximately 32 hours to train for 600 epochs on an Nvidia RTX 2080ti GPU with 11GB RAM, Intel i9 9900k CPU, and 32GB RAM. It is however possible to get near nearly as good performance in less time and epochs. Figure 7 shows that the training and validation loss does not decrease significantly after 150-250 epochs.

5.2 Overall Accuracy

We start by looking at the overall accuracy for the five different models used. This is shown in Table 6.

Model	MSE	MAE	MAPE
Baseline	499.537	4.422	35.051
M1	46.317	3.128	27.903
M2	40.982	2.897	25.585
M3	40.922	2.884	25.339
M4	40.047	2.864	25.289

Table 6: Models Performance on Test Set

From Table 6 it is clear that all BPNN model (M1-M4) outperform the baseline model. The BPNN models incrementally get more accurate, i.e., lower MSE, MAE, and MAPE, as a larger feature set is used, see Table 5.

The model M1 only uses three features and is clearly outperforms the baseline model on all three accuracy metrics. However, the baseline model is not a naive dummy model as the next subsections will show.

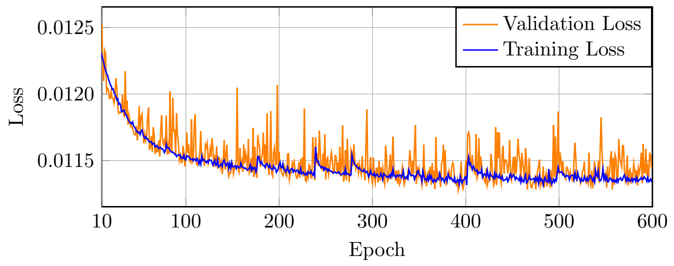


Figure 7: Training and validation loss of M4

In M2, we do not know how much the tortuosity and one-way features individually affect model accuracy. Figure 6 shows that most road segments have a tortuosity near 1.0. Because 4.6% of road segments and 24.9% of the GPS data rows are from one-way roads this feature is likely to affect the model more than tortuosity.

The minor improvement from M2 to M3 can be attested to the relatively small number of road segments that have the annotations speed bumps or pedestrian crossings. 0.21% of road segments have speed bumps and 0.45% of the rows are on these segments. Similarly, 0.06% of road segments have pedestrian crossings and 0,1% of the rows are on these segments.

However, the model M3 is still capable of using this limited additional data to make a model that consistently outperforms the model M2. These annotations, therefore, affect the travel time. Being able to show this small effect is also an incitement to contribute such annotation to for example OpenStreetMap to improve the accuracy of travel-time predictions.

The model M4 uses the most features and is consistently the model that performs the best. The improvement in accuracy over the other BPNN model is small and is most likely due that the additional features make the feature space too big relative to the amount of training data, for the model to learn without more data. More on this in subsection 5.5.

5.3 Road Category Accuracy

We raise the granularity and examine the accuracy at a road-category level in this section.

Table 7 shows the MAPE per road category for the five models presented in section 4. The MAPE is chosen to be able to compare accuracy results across different categories.

For the *motorway* category, all models work well with a MAPE as low as 10.386. The models M1-M4 also work well on the *motorway_link* category. Here the baseline model is significantly more inaccurate. For *motorway_link* in general, the results are less accurate than for the *motorway* category because the speed varies more, i.e., drivers are speeding up to enter a motorway or slowing down to leave the motorway.

For the *motorway* and *motorway_link* categories, there are no speed bumps or pedestrian crossing. This is a likely reason why M3 is less accurate than M2 on the *motorway_link* category. The results for *trunk* and *trunk_link* categories are very similar to their motorway equivalent.

The results in Table 7 show that for road categories where the traffic flows in the same direction, e.g., *motorway* and *trunk*, the travel time is predicted more accurately. This is a likely explanation to that the estimates on the *primary* road category is lower than for *motorway* and *trunk*.

The accuracy on the *secondary* and *tertiary* road categories are lower than for the *primary* category. This is most likely due to higher variance.

For all five link road categories, the MAPE is significantly higher compared to their non-link counterparts. This is likely due to the acceleration associated with link categories, i.e., the acceleration to enter or leave a motorway. Figure 12 shows that the number of road segments on the five link road categories are very low. The lower accuracy on these categories is therefore considered a minor issue.

5.4 Route Accuracy

To evaluate the accuracy experiences by the drivers, the travel times of 18 routes are inferred with each of the different models. The

Model	motorway	motorway_link	primary	primary_link	residential	secondary	secondary_link	tertiary	trunk	trunk_link	unclassified	tertiary_link
Baseline	12.787	134.841	27.648	36.435	34.898	34.085	35.607	39.938	15.429	27.244	40.524	45.709
M1	10.574	17.108	23.326	22.999	34.676	28.824	26.419	29.209	14.174	17.048	30.591	66.612
M2	10.812	15.285	18.350	23.306	33.764	25.859	28.278	27.486	**12.170**	**14.816**	**28.405**	49.005
M3	10.641	15.326	18.142	**20.782**	**32.957**	25.453	34.168	27.346	13.106	14.987	29.026	56.964
M4	**10.386**	**15.235**	**17.991**	23.284	33.568	**25.381**	**24.541**	**27.249**	12.928	15.003	28.413	**40.236**

Table 7: MAPE per Road Category

routes are described in Table 8 and shown on the map in Figure 8. The routes are both in city and rural areas to examine if the models presented in this paper can be used in city areas.

No	Segs	Len	Min.	Type	Categories
1	6	7.87	2000	Rural	sec.
2	4	2.15	7000	Rural	sec.
3	51	24.12	450	Rural	pri. trunk
4	23	4.87	25	Rural	sec.
5	14	8.08	400	Rural, Urban	trunk
6	178	32.24	400	Rural, Urban	sec. un. res.
7	65	16.84	60	Rural, Urban	pri.
8	2	8.65	99	Rural	mot.
9	20	4.13	800	Urban, Rural	pri. ter.
10	6	1.14	45	Rural	pri.
11	11	5.03	450	Rural, Urban	sec.
12	6	16.09	825	Rural	mot.
13	23	17.06	10	Rural	pri. trunk
14	19	37.70	1400	Urban, Rural	sec.
15	6	0.73	1300	Urban	res. un.
16	7	2.13	1900	Rural, Urban	sec.
17	21	9.17	800	Rural	pri.
18	6	1.09	60	Urban	pri.

Table 8: Benchmark Routes

The ground truth travel time of each route is the mean travel times of a minimum of 10 cars that have traveled the same exact routes using a *strict path query* [10]. Please note that this is a one-to-one comparison as the ground-truth travel time is based on the non-aggregated GPS data used in this paper.

To infer the travel time of a route, each road segment of the route is extracted along with all features needed by the model. The route travel time is then the sum of all segment travel times.

To make predictions using the M4 model extra steps are required to get a travel time. M4 has the three temporal inputs month, weekday, and hour. In the benchmarks these inputs are unspecified. To get a general travel time that covers all months, weekdays, and hours, a prediction is made for each combination of the three features, which gives 2016 predictions per road segment. The road segment travel time is then the mean of all 2016 predictions. These extra steps only have a minor overhead due to the inputs and model fitting into a few or a single GPU batch.

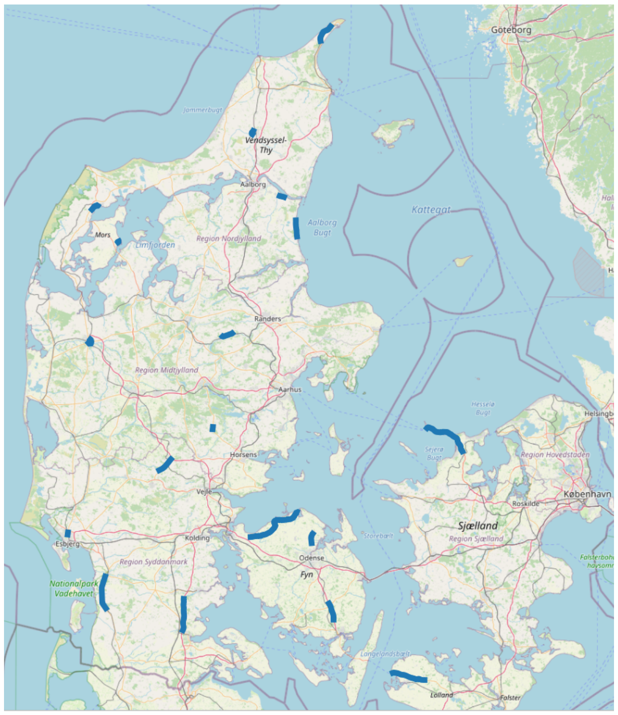


Figure 8: Benchmark Routes Placement

Table 9 shows the results for the benchmark routes where the most accurate results are on a green background and the most inaccurate on an orange background. The column *GT* is the ground truth. The columns *CS1* and *CS2* are the travel time from two commercial solutions.

Consistent with the overall and road-category results model M4 is the most accurate in 10 cases. More surprisingly, the baseline model is the most accurate in 6 cases. Routes 1, 2, 4, and 12 are only in rural areas and can explain why the baseline model is the most accurate.

On the contrary, the baseline model is also the most inaccurate in 10 cases. In all these cases the baseline model always estimates a higher travel time than the models M1-M4 and the ground truth. This is due to speeding. Model M4 is the most inaccurate in 4 cases. In these cases, the estimates are considerably below the ground truth.

No	GT	BL	M1	M2	M3	M4	CS1	CS2
1	489	488	340	334	338	332	360	360
2	143	133	98	96	97	96	120	120
3	1029	1224	1071	1077	1079	1038	1140	1080
4	387	302	252	244	248	245	420	300
5	319	365	328	323	321	321	360	300
6	1593	2033	1723	1700	1720	1686	2100	2040
7	823	1030	847	840	845	819	840	840
8	243	260	264	258	260	255	240	240
9	515	493	335	335	336	328	300	300
10	44	62	54	52	52	51	60	60
11	208	312	230	225	227	226	240	240
12	377	485	497	485	493	486	540	420
13	796	864	770	761	763	754	1020	840
14	249	234	199	207	209	195	300	240
15	109	70	66	78	77	78	120	120
16	110	132	101	102	102	104	120	120
17	431	494	415	413	410	404	420	360
18	55	84	66	65	64	63	120	120

Table 9: Travel-Times in Seconds on Benchmark Routes

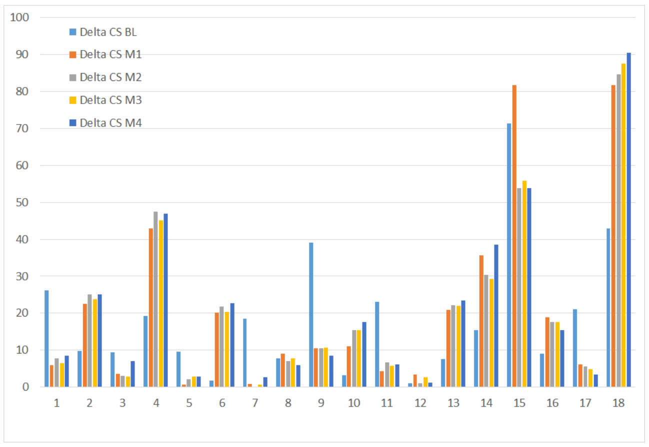


Figure 9: Difference Commercial Solutions and Models

In general, the M4 model predicts the lowest travel time on 13 of 18 routes. Further, the models M1-M4 predict lower travel times than the baseline model except for routes 8, 12, and 15. This indicates that many drives above the speed limit on most of the benchmark routes.

The percentage difference in travel time between the commercial solutions and the five models on the benchmark routes is shown in Figure 9. The commercial solutions report in minutes. This coarser granularity makes the differences to the predicted values very large for short routes such as routes 10, 15, 16, and 18. For 9 of the 18 routes the difference is below 10% and for 14 of 18 routes, the difference is below 25% for M4.

The model M4 can take month and hour-of-day into consideration as the only model presented in this paper. The variation in the travel time over months is shown in Figure 10 for the 18 benchmark routes. The routes are split into two based on travel time.

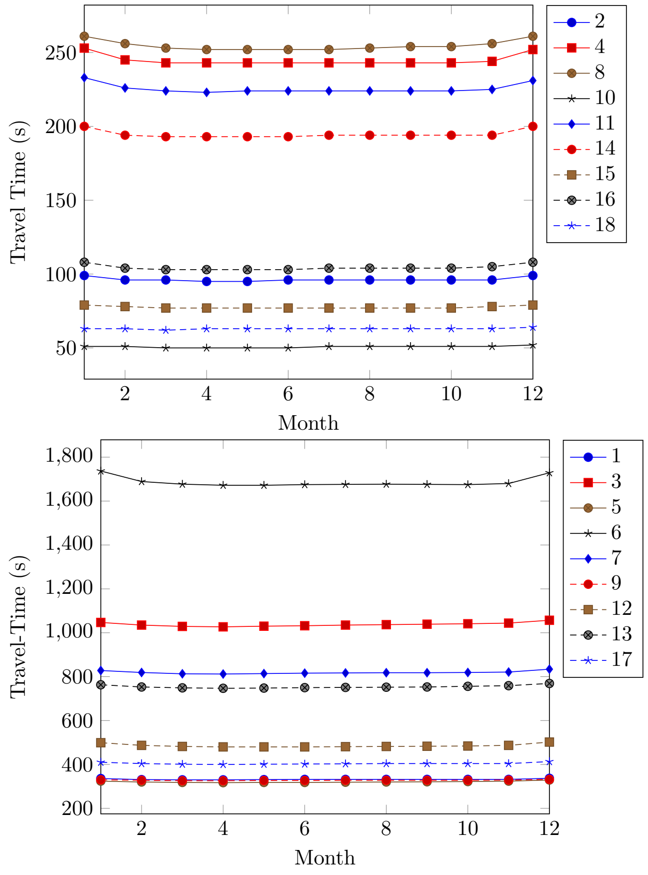


Figure 10: Travel-Time Estimation of Months of the Year

The travel-time estimation varies only a little over the year. However, it is slightly higher in the winter month. This is realistic as traffic is slightly slower due to ice and snow.

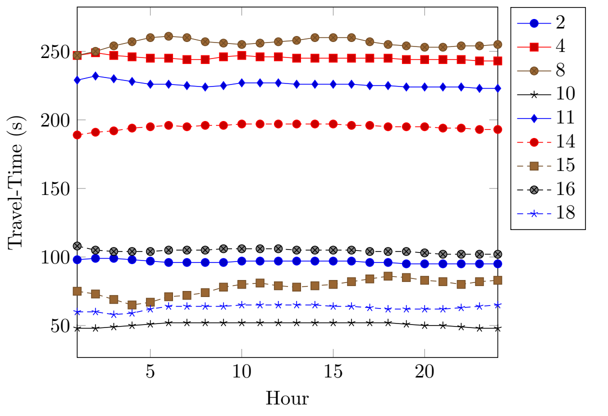
The variation in the estimated travel time over the day for workdays (Monday to Friday) is shown in Figure 11. Again the travel-time estimates are fairly stable. However, the travel time is on some routes a bit lower during the night, e.g., routes 3, 7, and 14.

Figure 10 and Figure 11 show that model M4 can use the variation in training GPS in the travel-time predictions.

5.5 GPS Dataset Size Influence on Accuracy

The final study of accuracy look at the relationship between model accuracy and the size of the GPS training dataset. We use training sets sizes of 100K, 500K, 1M, 5M, 10M, and the full dataset with 45.5M rows. The rows are picked randomly, however, we always add to the dataset such that 100K dataset is included in the 500K dataset, the 500K dataset in the 1M dataset, and so on. Figure 12 shows the distribution of road categories is represented in each subset.

The same feature sets are used as in models M1, M2, M3, and M4 in Table 5. We label the features sets A, B, C, and D respectively. The subsets are split into the training and validation set in a 3:1 split, to keep the same relative size between these sets. To relate the experiment to a real-world scenario, where more data is gathered over time the models are only trained for 8 hours each. To further



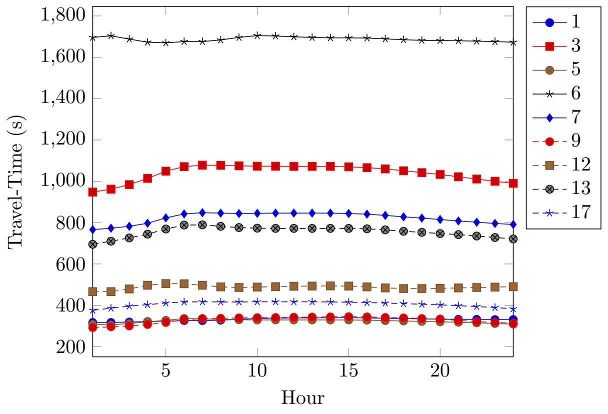


Figure 11: Travel-Time Estimation for Workdays

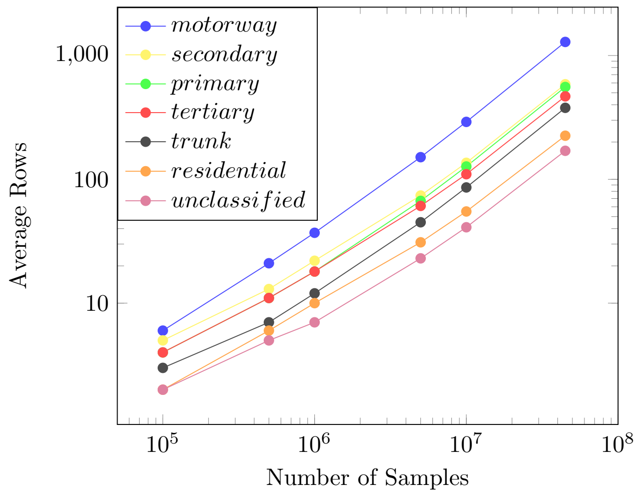


Figure 12: Average Number of Rows per Subset

save time, training of a model stops if the validation loss does not improve within 15,000 back propagations. By conditioning the early stopping on the number of back propagations, instead of a number of epochs with no improvement, the stopping condition is not biased towards large datasets.

For each training run only the model state with the lowest validation loss is saved. Large datasets have more batches per epoch, and therefore require more back propagations per epoch, due to the batch size of 70,000. For example, the 100K samples subset only has one backpropagation per epoch, while the full dataset requires 480 back propagations per epoch.

The results in Figure 13 show that 100,000 training samples is enough to outperform the baseline model when comparing to the overall accuracy shown in Table 6.

For the simplest model with feature set A 100K samples are sufficient and this model does not improve as the number of samples increases. Models with feature sets A and B uses five and seven features respectively. These models benefit from more GPS data samples but only up to 1M samples. Adding more GPS data samples does not improve these models. The most complex model with feature set D that contains all ten features benefits the most as more and more GPS data is available. This model is most accurate when trained using the full GPS dataset. The improvements are the largest when going from 1M to 5M samples. The 10M and full dataset are better, but the improvement is not as significant. Models with feature set D and 100k to 5M samples have lower accuracy than with feature sets B and C. This is likely because the feature space becomes too large for the models to learn without more data. Only the full data set has enough data to give a better model. The graphs indicate that only feature set D might give an even better model with more data and.

The runtime reveals which models stopped training because they are overfitted. Where the runtime is eight hours the model are most likely not overfitting, and where the runtime is lower the model training stopped early because it is overfitted. For feature set D the models overfit within the eight hours, but at an increasing number of hours as the dataset size increase, except for when trained on the full dataset.

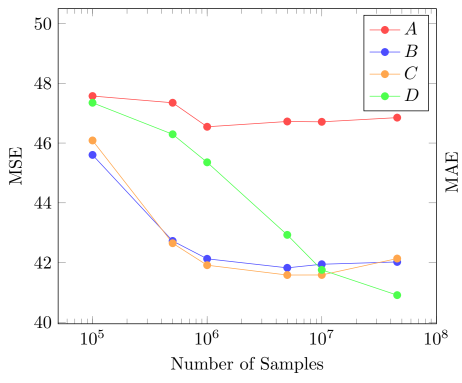
6 CONCLUSION

The main hypothesis of this paper is that travel time can be estimated accurately in rural areas because there is limited variation in the travel time over the day and because there are no issues with congestion or signalized intersections. We show that a simple baseline model using the speed limit can be the most accurate on a route. However, the baseline model is generally outperformed by the neural-network models going from a MAPE of 35.051% to 25.289%.

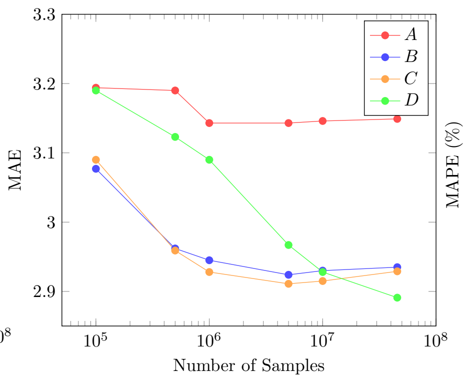
We show that the accuracy of the travel-time predictions can be improved by adding road network annotations for speed bumps and pedestrian crossing. Even though these annotations are not complete and only one a small fraction of road segments. This can encourage municipalities to contribute to map solutions such as OpenStreetMap.

The inclusion of travel-time information from GPS dataset increases the accuracy of travel-time predictions. The simplest models only require 100K samples whereas the most complicated models continue to improve with a maximum dataset of 45.5M samples. However, there are diminishing returns in adding larger GPS datasets.

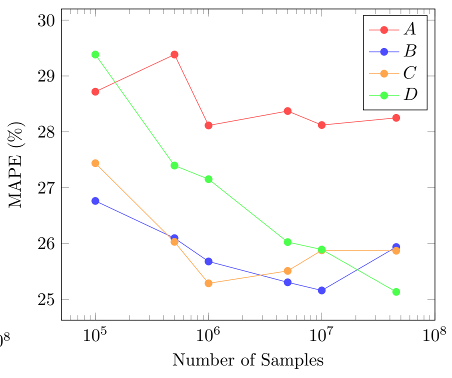
Overall, the solutions provided in this paper can be used as an alternative in rural areas with limited congestion problems. If municipalities contribute to road-network annotations, e.g., speed



(a) MSE



(b) MAE



(c) MAPE(%)

Figure 13: Metrics Over Sample Size

bumps and pedestrian crossings the solutions can increase the accuracy of travel-time estimates.

Future work includes adding road slope as a feature and using features from neighbor segments in the road network. Naturally extending the work to cover city areas is a very interesting research direction.

ACKNOWLEDGEMENT

This work is supported by the EU Interreg RESPONSE project response-project.eu.

REFERENCES

[1] Sofiane Abbar, Rade Stanojevic, and Mohamed Mokbel. 2020. STAD: Spatio-Temporal Adjustment of Traffic-Oblivious Travel-Time Estimation. In *2020 21st IEEE International Conference on Mobile Data Management (MDM)*. 79–88. https://doi.org/10.1109/MDM48529.2020.00029

[2] Ove Andersen and Kristian Torp. 2017. Sampling Frequency Effects on Trajectory Routes and Road Network Travel Time. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (Redondo Beach, CA, USA) *(SIGSPATIAL '17)*. Association for Computing Machinery, New York, NY, USA, Article 30, 10 pages. https://doi.org/10.1145/3139958.3140024

[3] Lukas Biewald. 2020. Experiment Tracking with Weights and Biases. https://www.wandb.com/ Software available from wandb.com.

[4] J. Brownlee. 2018. *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*. Machine Learning Mastery. https://books.google.dk/books?id=T1-nDwAAQBAJ

[5] Che-Man Cheng and Xiao-Qing Jin. 2018. *Matrix Decomposition*. Springer New York, New York, NY, 1280–1290. https://doi.org/10.1007/978-1-4939-7131-2_153

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[7] Chenjuan Guo, Bin Yang, Jilin Hu, and Christian Jensen. 2018. Learning to Route with Sparse Trajectory Sets. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. 1073–1084. https://doi.org/10.1109/ICDE.2018.00100

[8] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[9] Xiangjie Kong, Menglin Li, Kai Ma, Kaiqi Tian, Mengyuan Wang, Zhaolong Ning, and Feng Xia. 2018. Big Trajectory Data: A Survey of Applications and Services. *IEEE Access* 6 (2018), 58295–58306. https://doi.org/10.1109/ACCESS.2018.2873779

[10] Benjamin B. Krogh, Nikos Pelekis, Yannis Theodoridis, and Kristian Torp. 2014. Path-based queries on trajectory data. In *Proceedings of the 22nd ACM SIGSPATIAL*. ACM, 341–350. https://doi.org/10.1145/2666310.2666413

[11] X. Li, F. Li, X. Dai, and H. Liang. 2019. A Matrix-Decomposition-Based Context Tensor Approach for Personalized Travel Time Estimation. In *2019 Seventh International Conference on Advanced Cloud and Big Data (CBD)*. 258–265. https://doi.org/10.1109/CBD.2019.00054

[12] Yang Li, Dimitrios Gunopulos, Cewu Lu, and Leonidas Guibas. 2017. Urban Travel Time Prediction Using a Small Number of GPS Floating Cars. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (Redondo Beach, CA, USA) *(SIGSPATIAL '17)*. Association for Computing Machinery, New York, NY, USA, Article 3, 10 pages. https://doi.org/10.1145/3139958.3139971

[13] Xi Lin, Yequan Wang, Xiaokui Xiao, Zengxiang Li, and Sourav S. Bhowmick. 2019. Path Travel Time Estimation Using Attribute-Related Hybrid Trajectories Network. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 1973–1982. https://doi.org/10.1145/3357384.3357927

[14] Ilya Loshchilov and Frank Hutter. 2017. Fixing Weight Decay Regularization in Adam. *CoRR* abs/1711.05101 (2017). arXiv:1711.05101 http://arxiv.org/abs/1711.05101

[15] Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

[16] Kun Tang, Shuyan Chen, and Aemal J. Khattak. 2018. Personalized travel time estimation for urban road networks: A tensor-based context-aware approach. *Expert Systems with Applications* 103 (2018), 118 – 132. https://doi.org/10.1016/j.eswa.2018.02.033

[17] Kun Tang, Shuyan Chen, Aemal J Khattak, and Yingjiu Pan. 2019. Deep Architecture for Citywide Travel Time Estimation Incorporating Contextual Information. *Journal of intelligent transportation systems* (2019), 1–17.

[18] Hongjian Wang, Yu-Hsuan Kuo, Daniel Kifer, and Zhenhui Li. 2016. A Simple Baseline for Travel Time Estimation Using Large-Scale Trip Data. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (Burlingame, California) *(SIGSPACIAL '16)*. Association for Computing Machinery, New York, NY, USA, Article 61, 4 pages. https://doi.org/10.1145/2996913.2996943

[19] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, and Gao Cong. 2021. A Survey on Trajectory Data Management, Analytics, and Learning. *ACM Comput. Surv.* 54, 2, Article 39 (March 2021), 36 pages. https://doi.org/10.1145/3440207

[20] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel Time Estimation of a Path Using Sparse Trajectories *(KDD '14)*. Association for Computing Machinery, New York, NY, USA, 25–34. https://doi.org/10.1145/2623330.2623656

[21] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel Time Estimation of a Path Using Sparse Trajectories. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14)*. Association for Computing Machinery, New York, NY, USA, 25–34. https://doi.org/10.1145/2623330.2623656

[22] Robert Waury, Jilin Hu, Bin Yang, and Christian S. Jensen. 2017. Assessing the Accuracy Benefits of On-the-Fly Trajectory Selection in Fine-Grained Travel-Time Estimation. In *2017 18th IEEE International Conference on Mobile Data Management (MDM)*. 240–245. https://doi.org/10.1109/MDM.2017.40

[23] Nikolaos Zygouras, Nikolaos Panagiotou, Yang Li, Dimitrios Gunopulos, and Leonidas Guibas. 2019. HTTE: A Hybrid Technique For Travel Time Estimation In Sparse Data Environments. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (Chicago, IL, USA) *(SIGSPATIAL '19)*. Association for Computing Machinery, New York, NY, USA, 99–108. https://doi.org/10.1145/3347146.3359096