# A Study on ETA Prediction using Machine Learning and Recovered Routes

Arjanit Arifi[1], Panagiotis Bouros[1] and Theodoros Chondrogiannis[2]

[1]Institute of Computer Science, Johannes Gutenberg University Mainz, Germany
[2]Department of Computer and Information Science, University of Kostanz, Germany

**Abstract**

The problem of finding the Estimated Time of Arrival (ETA) for a given vehicle finds several applications in scenarios such as public transport and car navigation. Recent advances in machine learning have significantly improved ETA models, resulting in more precise travel time predictions. However, the scarcity of comprehensive data sets that contain complete trajectories for training these models poses a challenge. To address this limitation, we consider route recovery as an alternative. We conduct a case study to explore the feasibility of leveraging recovered routes as input for machine learning-driven ETA models and evaluate their performance. To the best of our knowledge, this is the first study that considers such a setting for ETA prediction. Our analysis considers multiple machine learning models, namely statistical, tree-based ensemble, and deep-learning models. Our experiments reveal that the accuracy of the tree-based and deep learning ETA models on recovered routes is heavily affected by the methodology used to estimate the travel time attribute as a target variable for ETA prediction.

**Keywords**

Estimated time of arrival, Machine learning, Route recovery, Road networks

## 1. Introduction

Estimating the *time of arrival* (ETA) on a road network plays a key role for modern smart cities. In public transportation, predicting a reliable and accurate arrival time for buses and taxis, is critical as passengers often have time-sensitive appointments or require flexible transportation connections. Private cars rely on navigational services, e.g., on smartphones, and ETA prediction for everyday commuting or trip planning. Modern logistics services need reliable ETA prediction to offer real-time delivery tracking. The proliferation of GPS-enabled and sensor devices has contributed to the availability of real-time traffic information. Under this premise, ETA methods can adjust in real-time their prediction in case of traffic congestion or other unexpected events, while drivers have the opportunity to react earlier to changes and consider alternative routes.

Besides real-time traffic information, historical moving data can also be used to determine a reliable ETA, especially since traffic volume and travel speed tend to vary based on the time of day, day of the week, and weather. These features must be considered to predict an accurate ETA, so traffic patterns are learned, and the model can predict the travel time for a given trip, at a particular time. For example, if a driver is expected to drive over a heavily trafficked road in the next hour, this must be reflected in the ETA prediction, even if there is currently no traffic jam on that road.

Over time, various methods have been developed to train ETA models using historical data. Nowadays, the focus is primarily on Machine Learning (ML) methods, which are also utilized in navigation and taxi services such as Google Maps and Uber [1, 2]. ML-driven ETA models receive as input, a source and a destination location in the road network along with the departure time, and output the estimated time of arrival. For this purpose, ML models are trained offline using historical trajectories, travel times and other relevant features. During this training phase, such models learn the relationships between features and travel time, enabling them to make ETA predictions based on the learned dependencies.

**Motivation**. The ability of ML models to learn complex dependencies varies on the specific learning algorithm employed. Deep learning models, in particular, require large training data sets to recognize complex spatio-temporal dependencies between features [3]. Such data sets typically comprise a set of historical trajectories represented as a sequence of GPS-points, often collected by portable GPS-devices. However, publicly accessible trajectory data sets with comprehensive information are rarely readily available due to privacy concerns [4] or simply because of high purchase costs. Instead, only trip data are published, i.e., source and destination locations, start and end timestamps, e.g., in https://www.nyc.gov/site/tlc/index.page.Furthermore, even for published trajectories, there exists the problem of low sampling rate, which results in many consecutively sampled locations being far apart from each other. In such cases, it is unclear which route out of several possible ones the vehicle did actually follow [5]. As a result, such trajectory data sets are too sparse to sufficiently train ML-driven ETA models, especially when the complete trajectory is unavailable.

To deal with the limited availability of historical trajectories, *route recovery* has attracted significant attention in the last decade. Given the source location, the destination location and the duration of a trip, the route recovery problem aims at determing the actual route followed as accurately as possible. The majority of existing route recovery solutions either rely on map-matching and therefore aim at recovering routes based solely on empirical observations [6, 7, 8], or utilize historical information, i.e., past trajectories [9, 10]. Different from the previous efforts, the recent study by Chondrogiannis et al. [11] presented recovery techniques in the absence of historical data, which rely on traversing the road network.

**Contributions**. We conduct a case study to investigate the role of route recovery on training ML-driven models for ETA prediction when historical trajectory data are unavailable. To our knowledge, this is the first work to consider such a setting. Our contributions are summarised as follows:

*Published in the Proceedings of the Workshops of the EDBT/ICDT 2024 Joint Conference (March 25-28, 2024), Paestum, Italy*
ararifi@students.uni-mainz.de (A. Arifi); bouros@uni-mainz.de (P. Bouros); theodoros.chondrogiannis@uni.kn (T. Chondrogiannis)
0000-0002-8846-4330 (P. Bouros); 0000-0002-9623-9133 (T. Chondrogiannis)

- We devise a novel workflow which uses recovered routes to train ETA models. We enumerate the features employed by the models and how these features are determined for the recover routes, instead of actual trajectories.
- We consider three ML-driven models for ETA prediction that rely on different learning methods: a statistical, a tree-based ensemble, and a deep learning model.
- We consider the recovery methods recently proposed in [11], to recover a single route for every training trip.
- We conduct an experimental analysis to study the merit of route recovery for ETA prediction, in the city of Porto. We compare the accuracy of the ML models prediction when trained by a typical set of historical trajectories and when there are trained by recovered routes.

**Outline**. The remaining text is as follows. Section 2 briefly overviews the related work in ETA prediction and route recovery. Section 3 details the workflows compared in our case study. Section 4 describes the setting of our study and Section 5 reports our experimental findings. Last, Section 6 concludes our study with directions for future work.

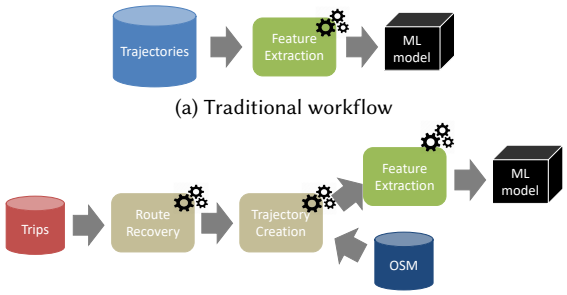
## 2. ETA and Route Recovery

We first discuss ML-driven models commonly employed for estimating the time of arrival (ETA) or travel time (TTE). These models are categorized into *statistical*, *tree-based ensemble*, and *deep learning*. Then, we briefly describe the single route recovery methods presented in [11].

### 2.1. ML Models for ETA

#### 2.1.1. Statistical Models

Statistical models typically consider historical data and statistical techniques to make predictions. The simplest approach is to use the speed limit on the road segments and the distance to the destination for ETA calculation [12]. However, this approach does not consider traffic conditions on road segments and is therefore not accurate. Additionally, the the actual speed of the vehicles is usually lower than the speed limit, especially in urban areas, due to traffic lights, traffic jams, etc. To take the traffic conditions into account, some approaches consider the Historical Average Speed (HAS) on the road segments. For instance, Maiti et al. [13] use bus trajectory data from a city in India to calculate HAS on a 15-minute time interval by taking the median of the speeds of all buses that passed every road segment during that time interval. Then to predict ETA, the distance to the destination is divided by HAS on the involved road segments. The authors compare the results to ML-models using Neural Networks (NN) and Support Vector Machines (SVM), to show that the statistical model performs similar to other models, but with significantly lower training and testing times. Further, Mang et al. [14] and Al-Naim and Lytkin [12] consider HAS-based prediction as a baseline. While HAS does offer a simple and fast baseline for ETA prediction, it fails to fully capture the traffic conditions on a trip. Such conditions are not only influenced by the time of the day, but also by the day of the week, the weather conditions, and other factors.




(a) Traditional workflow

(b) Workflow with recovered routes

**Figure 1:** ETA prediction workflows

#### 2.1.2. Tree-based Ensemble Models

Under this category, tree-based models are optimized by ensemble methods. Essentially, a decision tree represents a model that predicts the target variable (in our case, the ETA) based on the decision rules of the tree. These decision rules are learned during the training phase by finding the best split of the data at each node, such that the entropy of the target variable is minimized [15]. While decision trees are easy to interpret and perform well even with noisy data, they tend to over-fit. To avoid over-fitting, ensemble methods are used to combine multiple decision trees in order to reduce the variance of the model, e.g., random forest regression and gradient boosting. Gupta et al. [16] implemented and tested both methods, using a dataset of taxi trajectories, to predict TTE. The analysis showed that both methods perform well in terms of Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), even if only the source and destination of the trips are used as one of the features. The model performance is also optimized by tuning the hyperparameters of the models in order to minimize the Standard Deviation of the prediction error. Huang et al. [17] compared several types of tree-based models considering a dataset of over 9 million taxi trips, again for TTE prediction. They divided the travel time prediction into two distinct problems: long-term forecast (over several days) and short-term forecast (over the next hour). The tests indicated that, for long-term forecasts, Gradient Boosting and Random Forests outperform simpler models like Decision Trees in terms of RMSE. For short-term forecasts, the authors investigate the amount of training data required to make reliable predictions for the next hour. They discover that the RMSE remains relatively constant for different amounts of training data and that training the respective model within a 1-hour window is sufficient.

#### 2.1.3. Deep Learning Models

Tree-based ensemble models are able to capture spatio-temporal relationships between features and other factors such as weather conditions [17], but they still face challenges in capturing complex driving behaviors, such as turns, and traffic conditions between consecutive road segments. To address this issue, several works have utilized neural networks for ETA or TTE prediction. Amita et al. [18] employ a feed-forward neural network to train and test on a dataset of 40 bus trips in Delhi, India. Input features include the number of passengers boarding and alighting at each stop and the number of bus stops. After hyperparameter tuning, the neural network model's performance is compared to linear regression as a baseline. Results show that

the neural network model outperforms linear regression in terms of RMSE and MAPE. However, the authors note that using a larger dataset for training and validation could further enhance the results. Also, the features used may not sufficiently capture traffic conditions and other factors influencing travel time.

Wang at al. [19] introduced a more sophisticated deep neural network framework for TTE, called DeepTTE. DeepTTE combines a geo-spatial convolutional neural network, which captures spatial dependencies among road segments, with a recurrent neural network, which captures temporal dependencies. While convolutional neural networks are typically employed for image recognition, the authors adapt this concept to develop a network that effectively models the geo-spatial dependencies of trips. The training process splits the Travel Time Estimation problem into two subproblems: individual TTE prediction and collective TTE prediction. The individual TTE prediction accumulates the travel time of each road segment, while the collective TTE predicts the travel time for the entire trip directly. Each sub-problem has its advantages and disadvantages, as the individual prediction captures traffic conditions for individual road segments but not the inter-segment conditions (e.g., traffic lights at intersections). Conversely, the collective prediction captures overall traffic conditions but may be less accurate for long trips that traverse unobserved road segments. To overcome the drawbacks of each sub-problem, the authors concurrently train them for each trip using an attention mechanism. This allows the model to learn the optimal combination of both approaches by adjusting the weights of the sub-problems. Trained on two datasets with over 9M taxi trips in Chengdu and over 3M trips in Beijing, DeepTTE outperforms other existing models, including Gradient Boost Regression and Recurrent Neural Networks, in terms of MAPE, RMSE, and MAE. The authors also note that besides capturing spatio-temporal dependencies, the model can also incorporate other factors such as weather conditions or driver IDs to further improve predictions.

### 2.2. Route Recovery

#### 2.2.1. Route Recovery for Map-matching

Most existing methods for map-matching focus on the spatial similarity of a trajectory with the edges of the road network [20]. However, when the input trajectory is sparse, i.e., consecutive coordinates are far apart, most map-matching methods fail to find a matching route. To address this problem, many map-matching algorithms simply compute the fastest path between coordinates that are far apart [21, 22, 23]. However, since one cannot expect drivers to always choose the fastest path [24], other approaches compute paths that minimize costs determined based on empirical observations. Zheng et al. [25] use geometric and topological information of the road network to determine edge weights. Rahmani and Koutsopoulos [26] infer edge weights using a heuristic function that considers delays at traffic lights and left turns.

#### 2.2.2. Route Recovery using Historical Data

Methods that utilize historical trajectories usually train a machine-learning model to learn spatial transition probabilities. Jagadeesh and Srikanthan [27] employ a hidden Markov model that learns transition patterns. Zheng et al. [28] model the spatial transition probability between adjacent edges in a road network with one-order Markov model. Banerjee et al. [29] use Gibbs sampling, in which the spatial transition probabilities are modeled using high-order Markov chains. Wu et al. [10] employ inverse reinforcement learning to capture the spatial transition patterns. Due to the fact that we do not assume the availability of historical trajectory data, the aforementioned works are not applicable on our problem setting.

## 3. ETA Prediction Workflows

The process of training the learning models with the original trajectory data is straightforward, as shown in Figure 1a. All ML-based ETA prediction models accept a set of trajectories as input. Some models like DeepTTE require trajectory data to be on a very specific format, e.g., locations have to be sampled on fixed intervals, or additional information, e.g., whether a given day was a holiday and an id for every driver. Nevertheless, the training of a model relies mainly on the encoding of the sequence of the timestamped locations.

That is not the case with recovered routes. While it is possible to extract a sequence of coordinates from each recovered route, we still need timestamped locations to train the models. Further, many models require the time-gap between the sampled locations, as they assume the locations are sampled at a fixed rate. To address this issue, we designed a transformation framework that recovers routes from trips and then creates trajectories. Figure 1b illustrates our proposed workflow. The main difference to the traditional in Figure 1a lies in the *Route Recovery* and the *Trajectory Creation* components. Given a source and a target location, the starting timestamp and the duration of a trip, the *Route Recovery* component infers the most likely route that the driver used. Then, the *Trajectory Creation* extracts a trajectory by simulating the movement of the vehicle along the recovered route. Last, the computed trajectory is used as input to a ML-based ETA prediction model, similar to the traditional workflow.

## 4. Case Study

We next present our case study, first elaborating on its setup. We describe the dataset used, the route recovery methods, the trip features for training and the tested models.

### 4.1. Dataset

We used the Porto dataset[1] from the ECML/PKDD competition in 2015 [30], where the goal was to predict the destination of taxi trips based on existing partial trajectories. The dataset contains the trajectories of 1,710,670 trips recorded by 442 taxis in the city of Porto from July 1, 2013, to June 30, 2014. Figure 2 shows the heatmap of the trajectories projected on the city of Porto. Evidently, the trips extend beyond the city of Porto and therefore, we considered the entire road network of the Porto district, as opposed to only focusing on the city. We obtained the network from the OpenStreetMap project (OSM)[2]. It contains 4,963 nodes, which model road intersections, and 10,467 edges, which model road segments.

[1] https://kaggle.com/competitions/pkdd-15-predict-taxi-service-trajectory-i

[2] https://www.openstreetmap.org/

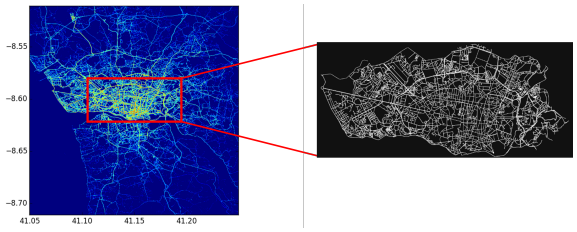



**Figure 2:** Heatmap of the Porto dataset (created using https://www.kaggle.com/code/mcwitt/heatmap/script)

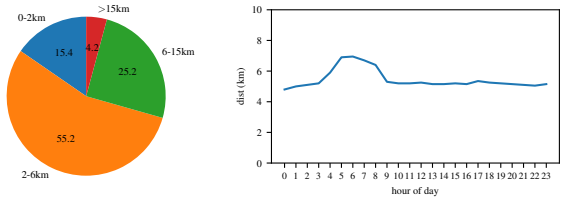



**Figure 3:** Distribution of trips distance in the Porto dataset

To conduct our analysis and obtain some of the features necessary for the models, we first map-matched the trajectories using Valhalla[3]. Due to the inherent noise in GPS coordinates and the inability of the map-matching engines to accurately reconstruct the exact route followed for some trips, there was a data loss of approximately 24%. As a result, we were able to process approximately 1,295,163 trajectories with their corresponding features after map-matching, out of which 80% (1,036,130) were used for training and 20% (259,033) for testing. The training and testing sets were randomly selected from the entire data set. The re-sampling step, which is necessary for training the deep learning models efficiently, resulted into an additional, but smaller, data loss. So, the training data set for the deep learning methods contains 1,029,333 trajectories and the test data set, 258,196.

Figure 3 elaborates on the distribution of the trip distances in the Porto dataset. The pie chart to the left reveals that trips of medium distance (2-6 km) account for over 55% of the dataset, while short trips ($<$ 2 km) and long trips (6-15 km) also have a considerable presence. Longer trips over 15 km constitute a smaller fraction, approximately 4%. These longer trips can still influence our ETA models as they often encounter multiple traffic conditions and traverse road segments that have not yet been observed (e.g., outside the city area). In addition, the plot to the right shows the average trip distance per hour of the day. The peak from 05:00 to 08:00 is caused by the morning rush hours when commuting to work takes place.

## 4.2. ML Models for ETA

To deliver a complete study, we implemented one representative model from each category in Section 2. Table 1 summarizes the dataset features used by each model.

### 4.2.1. Statistical Models

We implemented one statistical ETA prediction model, namely HAS. The model relies on the Historical Average Speed concept, similar to [12]. For each road segment and a specific time interval, HAS calculates the average speed of all vehicles that have traveled on that segment. Specifically, we first define 10-minute intervals and then create a mapping for each of them. This mapping assigns an average speed to every road segment traveled during a time interval. If a vehicle passes through a road segment within a certain time interval but no entry exists in the mapping, we create a new and store the average speed. If an entry already exists, we update the average speed using weighted exponential smoothing [14]:

$$V(t) = \alpha * v(t) + (1 - \alpha) * V(t - 1)$$

where $V(t)$ is the avg speed at time $t$, $v(t)$ is the avg vehicle speed, $\alpha$ is a smoothing factor and $V(t-1)$ is the historical avg speed on the road segment. We repeat the above for all trajectories or recovered routes in the training set.

### 4.2.2. Tree-based Ensemble Models

For tree-based models, we built upon the work of [16], which considered both Random Forest Regression (RFR) and Gradient Boost Regression (GBR). In our study, we include GBR as it has demonstrated better performance, and we utilize similar features as described in [16]. The only difference lies in the distance calculation. While the authors compute the Haversine distance between the source and the destination location, we use the exact distance of the trip that was calculated during the map matching process. This change is expected to improve the accuracy of the model since we no longer estimate trip distances.

### 4.2.3. Deep Learning Models

We considered DeepTTE [19], modified to save the best model with the lowest loss after the training. The Mean Absolute Percentage Error (MAPE) is used as the loss function. DeepTTE uses features that include not only the entire trajectory but also the time and distance between individual points (time_gap/dist_gap), as well as additional features, e.g., the DriverID and total distance. To enable appropriate data normalization, we store the means and standard deviations of these features in a configuration file.

## 4.3. Route Recovery

Following our previous work [11], we consider three approaches for the task or route recovery from the input trips.

### 4.3.1. Shortest/Fastest Path

The most straightforward way to recover a route for a given trip is to compute the *shortest path* (SP), i.e., the path with the lowest network distance. But for vehicle trajectories, the shortest path is not always the optimal path due to the various speed limits on the road. A more realistic approach is to consider the *fastest path* (FP) [21, 22, 23]. Assuming that drivers always abide by speed limits, the fastest path is the path that yields the lowest travel time between two locations under optimal conditions, i.e., no traffic.

### 4.3.2. Simplest Near-fastest Path

Our second approach is based on the work of Sacharidis and Bouros [31] on the computation of paths that are easy for drivers to follow. In particular, one of the presented problems deals with the computation of a route that is not much slower than the fastest path, but has low complexity,

[3]https://valhalla.github.io/valhalla/

**Table 1**
Trip features used for training by ML-based models.

| Feature | Description | HAS | GBR | DeepTTE |
|---|---|---|---|---|
| trip_ID | Unique identifier for trip | ✓ | ✓ | ✓ |
| driver_ID | Unique identifier for the driver | | | ✓ |
| polyline | List of coordinates (longitude, latitude) | ✓ | ✓ | ✓ |
| time | Travel time (in seconds) | ✓ | ✓ | ✓ |
| ts | Timestamp of the start of the trip (Unix Time) | ✓ | ✓ | ✓ |
| time_gap | Time gap between consecutive data points (in sec) | ✓ | ✓ | ✓ |
| road_ID | Road ID of road segment (matched to each data point) | ✓ | ✓ | |
| day_type | Day type of trip's start time (normal day, holiday, day before holiday) | ✓ | | ✓ |
| opath | OSM node ID-path of all nodes on the trip | ✓ | | |
| weather_ID | Identifier associated with weather conditions | | | ✓ |

i.e., involves a small amount of turns. More specifically, given a source $s$ and a target $t$, the *simplest near-fastest route* is the route that has the lowest number of turns among all near-fastest routes from $s$ to $t$. Following from [31], we consider the *near-fastest route with minimum turns* (Min-Turns) as a potential solution for the route recovery problem. During the route recovery process, the travel time of the near-fastest path computed by Min-Turns is bounded by the ground truth duration of the given query trip.

#### 4.3.3. Minimum Road Hierarchy Peaks

Our third approach for route recovery is based on the observation that road networks are usually characterized by an inherent hierarchical/highway structure [32]. During route planning, roads such as motorways that are multi-lane with high speed limits are preferred over, e.g., residential roads. This is because they allow faster traveling and are less likely to be affected by traffic [33]. However, to ensure optimality, the fastest path may contain switches from roads of higher priority to roads of lower priority. Hence, even though the fastest path yields the lowest travel time under optimal conditions, human drivers may prefer a slightly sub-optimal route. We assume that drivers choose a path that (1) is not much longer than the fastest path, (2) uses roads of high priority as much as possible, and (3) switches to roads with low priority only if necessary to reach their destination.

To quantify the above criteria, we measure the *road hierarchy peaks* in the road type hierarchy of a given route. A peak is defined as a sequence of two switches between road types of lower priority to higher priority and back. Based on the concept of road hierarchy peaks, we consider the *near-fastest path with minimum road hierarchy peaks* (Min-HP) as a potential solution for the route recovery problem. Similar to Min-Turns, during the route recovery, the travel time of Min-HP is bounded by the recorded duration of the query trip.

### 4.4. Model Training with Recovered Routes

After obtaining the sequence of nodes of the recovered route from the road network, we extract the road ID of each path. This allows us to calculate various features based on attributes such as node coordinates, road length, speed, and travel time. However, a challenge arises when dealing with timestamp features, such as the time of travel. These features cannot be readily estimated and are required for most ML-models. In such cases, we rely on the original dataset, which contains the timestamp features. These entries with timestamp features are then merged with the transformed routes based on the trip ID, resulting in a complete dataset with transformed routes and timestamp features.

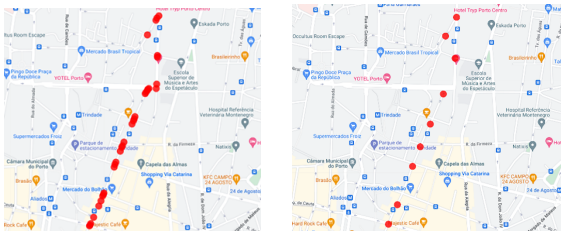



**Figure 4:** Re-sampling trip points

We are now able to train ML-models using the transformed recovered routes. However, we have observed that the recovered routes generally contain a much higher number of road IDs and consequently, coordinates per trip compared to the original trajectories. This is because the coordinates are stored at each node of the road network, without being sequenced based on time or distance. As a result, particularly in areas with many intersections, coordinates are generated that are very close to each other (around 10 meters). While this factor may not significantly impact statistical models or ensemble models due to sparse feature selection, it poses challenges for deep learning models. Due to the extremely close proximity of data points, neural networks and DeepTTE struggle to effectively learn from the data due to overly fine granularity. Additionally, DeepTTE assumes that the data points of trips should have a similar spacing. To address these issues, we implement a resampling step that takes a dataset of complete trips as input and adjusts arrays to ensure that each data point has an equal distance. Specifically, the arrays of latitude (lats), longitude (lngs), time gap (time_gap), distance gap (dist_gap), road IDs, and node path (opath) are resampled. In our case, we chose a resampling factor of 100 meters, resulting in the deletion of data points in between. With this resampling, we overcome the challenges posed by the high density of coordinates and create a more balanced and suitable dataset for training deep learning models, as shown in Figure4.

## 5. Experiments

We finally present the experimental results of our study.

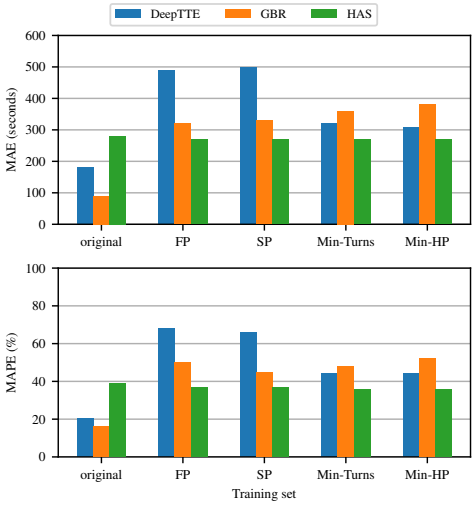



**Figure 5:** Comparing ETA models: different training datasets

## 5.1. Implementation Details

We implemented both training workflows in Python, using the pandas library[4] for data processing [34]. The code for the route recovery methods is publicly available by the authors of [11].[5] We also used the OSMnx[6] Python library to extract the road network of Porto district from OSM.

Similar to previous works, e.g., [12], we assess the accuracy of the ETA models on the test trips, by measuring the:

$$\text{Mean Absolute Error, } MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|$$

$$\text{Mean Absolute Percentage Error, } MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y_i}|}{y_i}$$

where $y_i$ denotes the real arrival time of the trip, and $\hat{y}_i$, the predicted one. Note that we used exactly the same test dataset for both workflows, which counts for the 20% of the Porto trajectories. For every test, we consider the source location and the destination, and the departure time.

We set the parameters for the ETA models as follows. In HAS, $\alpha$ is set to 0.5 for the exponential smoothing, i.e., new values contribute to the average speed of road segments with a weight of 50%. Similar to [16], we set `n_estimators` = 300, `random_state` = 19, `n_jobs` = -1 for GBR. For training and testing the models, we employ the methods from the scikit-learn[7] RandomForestRegressor and GradientBoostingRegressor classes (fit and predict) [35]. Last, for DeepTTE, we used the source code available by the authors[8], with `batch_size` = 256, `alpha` = 0.3, `kernel_size` = 3, and train the model for 3 epochs. We validate the model after each epoch and save the one with the lowest MAE.

## 5.2. Results

We compare the ML-driven models under the two training workflows in terms of their MAE and MAPE in Figure 5. When actual trajectories are used to train each model (group of bars termed "original"), we observe that the tree-based ensemble model GBR achieves the best accuracy. GBR yield the best result despite having sparse input features compared to deep learning models. In contrast, DeepTTE shows a slightly higher error that GBR, i.e., around 20% in MAPE. This finding contradicts the results in [19] but can be attributed to the difference in the sizes of the training datasets; Wang et al. [19] trained the model using over 9M trajectories while our training dataset contains only 1M, a number not large enough to accurately capture all spatio-temporal dependencies. Last, HAS is outperformed by the other models, showing a relatively high relative error slighlty below 40%, mainly because this model can capture traffic conditions only to a limited extent.

Now, when training the model with recovered routes (group of bars FP, SP, Min-Turns and Min-HP), we observe a different picture. Both DeepTTE and GBR exhibit a significant increase in their prediction error. Particularly noteworthy is DeepTTE for FP and SP, the MAPE of which exceeds 60%. In contrast, the accuracy of HAS remains relatively unaffected. As the statistical models require the fewer features to be trained, compared to tree-based ensemble and deep learning ones, they are less affected by the loss of information. Regarding the different route recovery methods, we observe some variations in their accuracy. For example, Min-HP and Min-Turns perform better for DeepTTE compared to the simple recovery methods FP and SP, while GBR demonstrates similar performance.

To gain more insight on our findings, we conducted extra tests to measure the accuracy of the models over the course of the day, and with respect to the characteristics of the test trips. In what follows, we focus only on the Min-HP recovery method, which exhibited the best results according to the analysis conducted in [11]. Figure 6 depicts the mean predicted travel time (i.e., the ETA minus the departure time) achieved by each model when trained with actual trajectories (top plot) and when trained by Min-HP recovered routes (bottom plot), from 00:00 to 23:00. We include in both plots also the real travel time of the test trips on average, for reference. We also report the MAPE in Figure 7. With the exception of HAS, we observe that when trained by actual trajectories, the predictions of the ML models are generally able to capture the variation of the travel time. Specifically, GBR exhibits the lowest MAPE, providing the most accurate predictions. DeepTTE predictions are less accurate but still, the model is able to capture the peaks throughout the day. In contrast, the predictions of HAS are as expected the least accurate. Moreover, the MAPE exhibits high variation as HAS is unable to capture how traffic conditions vary over the course of a day. The MAPE reaches the highest value from 09:00 to 20:00, a period which includes morning and evening rush hours; i.e., hours when a lot of people are on the roads and the traffic conditions are highly dynamic.

However, when the models are trained using Min-HP recovered routes, similar to Figure 5, we observe that both GBR and DeepTTE are significantly affected. Their MAPE rises higher than the MAPE of HAS as the latter is essentially unaffected by the different training dataset. If we juxtapose the predicted time of the models when trained by recovered routes to Figure 3 (right plot), we observe that this time follows the trend of the average covered distance per trip. All models incorporate the effect of the distance peak from 05:00 to 08:00, which results to their lowest MAPE values.

Last, we investigate the role of the trip time and distance in the accuracy of the models, again under the two different training workflows. Figures 8 and 9 report the results of our tests. The results paint a similar picture about how much

[4] https://pandas.pydata.org
[5] https://github.com/JohannBo/route-recovery
[6] https://osmnx.readthedocs.io/en/stable/
[7] https://scikit-learn.org/stable/
[8] https://github.com/UrbComp/DeepTTE

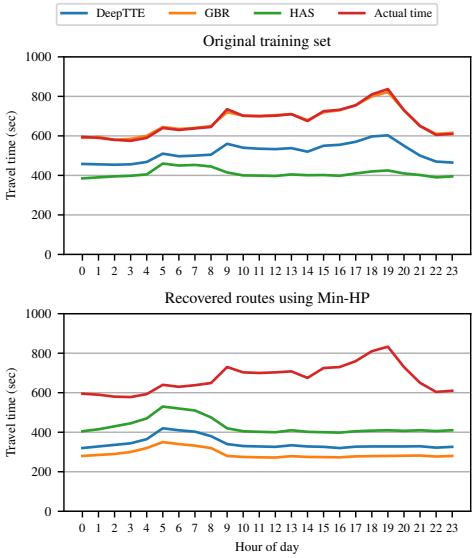



**Figure 6:** Comparing ETA models: mean travel time over hour of the day

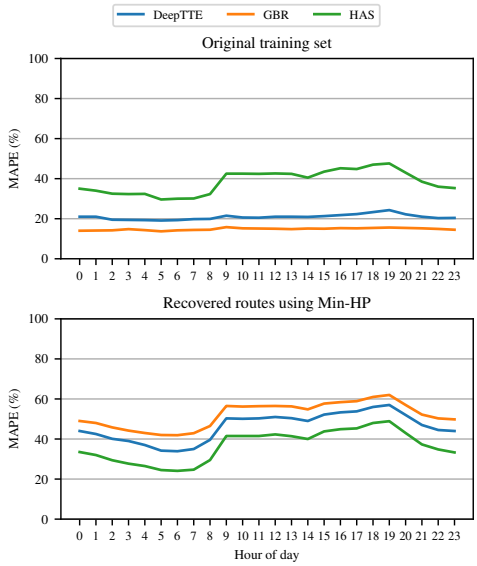



**Figure 7:** Comparing ETA models: MAPE over hour of the day

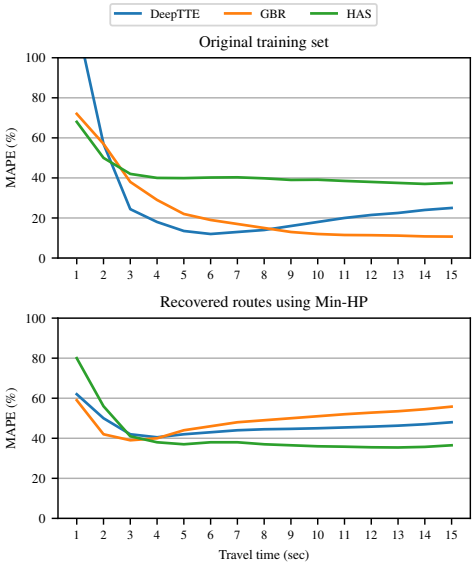



**Figure 8:** Comparing ETA models: MAPE over trip travel time

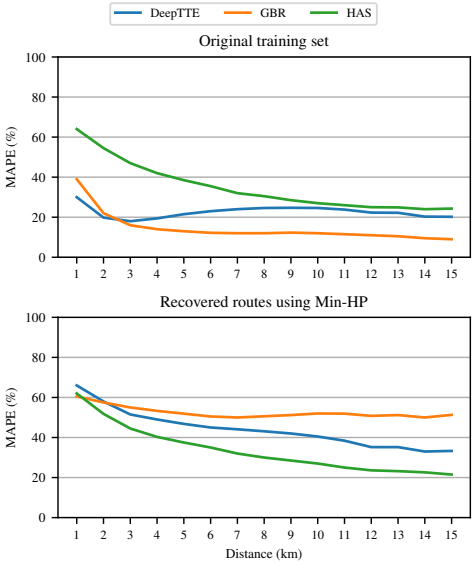



**Figure 9:** Comparing ETA models: MAPE over trip distance

the predictions of GBR and DeepTTE are affected when we train the models using recovered routes. However, we also notice that this impact is more pronounced for shorter trips in terms of both travel time and distance. As expected, in long trips especially in terms of covered distance, traffic delays and changes are related to a small part of the route and, therefore, driving decisions are overall less affected, compared to short trips when every traffic change will be directly reflected to the arrival time at the destination.

## 6. Conclusions and Future Work

In this paper, we evaluated the performance of three ETA prediction models by presenting a comprehensive framework that implements the two workflows. The first workflow involved training and evaluating various machine learning methods using original trajectories. In the second workflow, we generated recovered routes from the single-route recovery methods proposed. To feed this data into the ML-models, the data were first transformed, so that missing features like the distance of road segments were calculated or estimated from the road network. We compared the models from the two workflows and observed that recovered routes cannot reflect traffic conditions like original trajectories, resulting in the ML-models trained with recovered routes showing significantly poorer performance.

The result above highlights the need for further research towards improved route recovery methods. We plan to work towards this direction in the future. For instance, the study in [11] also introduced a region recovery task where a small subgraph of the road network that potentially contains the actual route, is returned, instead of a single predicted route. An interesting research question is how to train the ML-based models with recovered regions and whether this training approach has the potential to enhance the ETA prediction. As another direction for future work, we intend to investigate other prediction scenarios that require trajectories for training, e.g., traffic prediction [35, 36].

## Acknowledgments

T. Chodrogiannis was supported by the Deutsche Forschungsgemeinschaft (DFG) through Grant No. CH 2464/1-1. This work is based on the BSc thesis of Arjanit Arifi at Johannes Gutenberg University Mainz, Germany.

## References

[1] A. Derrow-Pinion, J. She, D. Wong, O. Lange, T. Hester, L. Perez, M. Nunkesser, S. Lee, X. Guo, B. Wiltshire, P. W. Battaglia, V. Gupta, A. Li, Z. Xu, A. Sanchez-Gonzalez, Y. Li, P. Velickovic, ETA prediction with graph neural networks in google maps, in: ACM CIKM, 2021, pp. 3767–3776.

[2] X. Hu, T. Binaykiya, E. Frank, O. Cirit, Deepreta: An ETA post-processing system at scale, CoRR abs/2206.02127 (2022). URL: https://doi.org/10.48550/arXiv.2206.02127.

[3] S. Wang, Z. Bao, J. S. Culpepper, G. Cong, A survey on trajectory data management, analytics, and learning, ACM CSUR 54 (2021) 1–36.

[4] M. Terrovitis, N. Mamoulis, Privacy preservation in the publication of trajectories, in: IEEE MDM, 2008, pp. 65–72.

[5] L. Lu, N. Cao, S. Liu, L. M. Ni, X. Yuan, H. Qu, Visual analysis of uncertainty in trajectories, in: PAKDD, 2014, pp. 509–520.

[6] W. Bian, G. Cui, X. Wang, A trajectory collaboration based map matching approach for low-sampling-rate GPS trajectories, Sensors 20 (2020) 2057.

[7] M. Srivatsa, R. K. Ganti, J. Wang, V. Kolar, Map matching: facts and myths, in: ACM SIGSPATIAL, 2013, pp. 474–477.

[8] C. Yang, G. Gidófalvi, Fast map matching, an algorithm integrating hidden markov model with precomputation, Int. J. Geogr. Inf. Sci. 32 (2018) 547–570.

[9] X. Li, G. Cong, Y. Cheng, Spatial transition learning on road networks with deep probabilistic models, in: IEEE ICDE, 2020, pp. 349–360.

[10] H. Wu, J. Mao, W. Sun, B. Zheng, H. Zhang, Z. Chen, W. Wang, Probabilistic robust route recovery with spatio-temporal dynamics, in: ACM SIGKDD, 2016, pp. 1915–1924.

[11] T. Chondrogiannis, J. Bornholdt, P. Bouros, M. Grossniklaus, History oblivious route recovery on road networks, in: ACM SIGSPATIAL, 2022, pp. 44:1–44:10.

[12] R. Al-Naim, Y. Lytkin, Review and comparison of prediction algorithms for the estimated time of arrival using geospatial transportation data, in: YSC, 2021, pp. 13–21.

[13] S. Maiti, A. Pal, A. Pal, T. Chattopadhyay, A. Mukherjee, Historical data based real time prediction of vehicle arrival time, in: IEEE ITSC, 2014.

[14] L. Meng, P. Li, J. Wang, Z. Zhou, Research on the prediction algorithm of the arrival time of campus bus, in: ITIM, 2017, pp. 31–33.

[15] T. M. Mitchell, Machine learning, International Edition, McGraw-Hill Series in Computer Science, McGraw-Hill, 1997.

[16] B. Gupta, S. Awasthi, R. Gupta, L. Ram, P. Kumar, B. Rohit Prasad, S. Agarwal, Taxi travel time prediction using ensemble-based random forest and gradient boosting model, in: ICBDCC, 2018, pp. 63–78.

[17] H. Huang, M. Pouls, A. Meyer, M. Pauly, Travel time prediction using tree-based ensembles, in: Computational Logistics, Cham, 2020, pp. 412–427.

[18] J. Amita, S. Jain, P. Garg, Prediction of bus travel time using ann: A case study in delhi, Transportation Research Procedia 17 (2016) 263–272.

[19] D. Wang, J. Zhang, W. Cao, J. Li, Y. Zheng, When will you arrive? estimating travel time based on deep neural networks, in: AAAI, 2018, pp. 2500–2507.

[20] P. Chao, Y. Xu, W. Hua, X. Zhou, A survey on map-matching algorithms, in: ADC, volume 12008, 2020, pp. 121–133.

[21] P. Newson, J. Krumm, Hidden markov map matching through noise and sparseness, in: ACM SIGSPATIAL, 2009, pp. 336–343.

[22] Y. Tang, A. D. Zhu, X. Xiao, An efficient algorithm for mapping vehicle trajectories onto road networks, in: ACM SIGSPATIAL, 2012, pp. 601–604.

[23] J. Yuan, Y. Zheng, C. Zhang, X. Xie, G. Sun, An interactive-voting based map matching algorithm, in: IEEE MDM, 2010, pp. 43–52.

[24] P. Cintia, M. Nanni, An effective time-aware map matching process for low sampling GPS data, CoRR abs/1603.07376 (2016). URL: http://arxiv.org/abs/1603.07376.

[25] Y. Zheng, M. A. Quddus, Weight-based shortest-path aided map-matching algorithm for low-frequency positioning data, Technical Report, 2011.

[26] M. Rahmani, H. N. Koutsopoulos, Path inference of low-frequency gps probes for urban networks, in: IEEE ITSC, 2012, pp. 1698–1701.

[27] G. R. Jagadeesh, T. Srikanthan, Robust real-time route inference from sparse vehicle position data, in: IEEE ITSC, 2014, pp. 296–301.

[28] K. Zheng, Y. Zheng, X. Xie, X. Zhou, Reducing uncertainty of low-sampling-rate trajectories, in: IEEE ICDE, 2012, pp. 1144–1155.

[29] P. Banerjee, S. Ranu, S. Raghavan, Inferring uncertain trajectories from partial observations, in: IEEE ICDM, 2014, pp. 30–39.

[30] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, L. Damas, Predicting taxi-passenger demand using streaming data, IEEE TITS 14 (2013) 1393–1402.

[31] D. Sacharidis, P. Bouros, Routing directions: Keeping it fast and simple, in: ACM SIGSPATIAL, 2013, pp. 164–173.

[32] P. Sanders, D. Schultes, Highway hierarchies hasten exact shortest path queries, in: ESA, 2005, pp. 568–579.

[33] C. S. Phibbs, H. S. Luft, Correlation of travel time on roads versus straight line distance, Medical Care Research and Review 52 (1995) 532–542.

[34] W. McKinney, Data structures for statistical computing in python, in: S. van der Walt, J. Millman (Eds.), SciPy 2010, scipy.org, 2010, pp. 56–61.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[36] D. A. Tedjopurnomo, Z. Bao, B. Zheng, F. M. Choudhury, A. K. Qin, A survey on modern deep neural network for traffic prediction: Trends, methods and challenges, IEEE TKDE 34 (2022) 1544–1561.