Path-based traffic flow prediction

Efstratios Karkanis[1], Nikos Pelekis[2], Eva Chondrodima[1] and Yannis Theodoridis[1]

[1]Department of Informatics, University of Piraeus, Piraeus, Greece
[2]Department of Statistics and Insurance Science, University of Piraeus, Piraeus, Greece

Abstract

Predicting traffic flow on a road network is crucial in various aspects of the transportation domain, encompassing safety and logistics. This paper introduces an innovative approach to forecast traffic flow, with a specific emphasis on predicting future traffic along paths within a road network. The proposed framework is tailored to accept GPS trajectory data as input, generating time series data illustrating traffic flow along designated pathways. It achieves this by utilizing only a subset of trajectories that strictly follow the paths without detours. This intuitive approach results in training data that better captures the essence of the forecasting problem. In the final step of our methodology, we employ state-of-the-art time series forecasting methods, including ensemble trees and recurrent neural networks. To validate the effectiveness of our approach, we evaluate it using a real-world dataset, demonstrating its capability in predicting traffic flow.

Keywords

Mobility, Trajectories, Traffic flow, Forecasting methods, Ensemble trees, Neural networks

1. Introduction

In the contemporary era, urbanisation has given rise to a conspicuous surge in the demand for public transportation within urban areas. This escalating need, fueled by population growth, economic advancement, and evolving lifestyle preferences, underscores a significant transformation in the dynamics of urban mobility [1]. Concurrently, challenges like traffic congestion, navigation optimisation, and air quality management have become prevalent[1]. As a result, path-based traffic flow, which in terms of this study refers to the number of vehicles passing through a particular path (a set of consecutive road parts between intersections) on a road network, has increased substantially over the recent decades, emphasising the critical necessity for accurate forecasting methods.

Mobility data analytics and prediction has gained an increasing interest in recent years, due to its time-critical applications in urban, maritime and aviation domains [2, 3, 4, 5]. In particular, predicting traffic flow is challenging due to its non-linear nature. This complexity arises because traffic flow is influenced by non-linear factors, such as weather conditions, road traffic accidents and public holidays, posing difficulty in accurate forecasting.

Typically, road networks are equipped with sensors that measure traffic flow at a specific point of the road network. On the other hand, specific vehicles, such as buses and taxis, are also expected to contain sensors that periodically transmit their mobility status information, including their GPS location, along with the respective sampling timestamp. The proposed framework aims to provide predictions for urban traffic flow along predefined paths by leveraging data collected from sensors embedded in specific vehicles, such as buses and taxis.

In summary, the complete workflow of the proposed framework involves several key steps. Starting with the original trajectory data provided by vehicles, we create a time series dataset that captures historical traffic flow values along predefined paths at uniform-sized time intervals. A key aspect of our methodology involves the measurement of traffic flow along each predefined path. Therefore, we perform a search procedure on trajectory data (the so-called Strict Path Query [6]) to retrieve all trajectories that strictly cross a predefined path of a road network within a specific time interval. A trajectory strictly follows a path if and only if this path is a subset of the trajectory. In other words, the trajectory does not deviate from this predefined path. In this way, we are sure that we compute traffic flow accurately, eliminating trajectories that perform detours while crossing the path. This approach allows us to reliably assess traffic flow throughout the entire length of the path, avoiding potential inaccuracies that might arise from considering trajectories with detours, which could lead to misleading indications of traffic flow.

Figure 1 illustrates a road network where letters a to l represent intersections. Four trajectories t_1, t_2, t_3, and t_4 are depicted, each marked with a distinct colour, navigating within the network. Additionally, there is a predefined path $b \rightarrow c \rightarrow h \rightarrow i \rightarrow d$, marked with orange colour. To assess traffic flow within this specified path

Published in the Proceedings of the Workshops of the EDBT/ICDT 2024 Joint Conference (March 25-28, 2024), Paestum, Italy
stratoskarkanis2@gmail.com (E. Karkanis); npelekis@unipi.gr (N. Pelekis); evachon@unipi.gr (E. Chondrodima); ytheod@unipi.gr (Y. Theodoridis)

[1]The European Green Deal: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2019:640:FIN

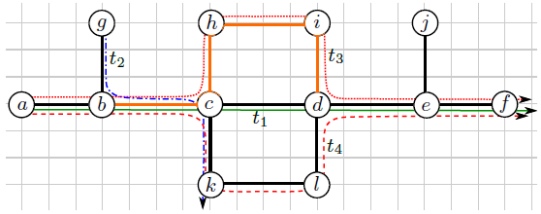

Figure 1: Road network with a predefined path coloured in orange and four navigating trajectories t_1, t_2, t_3 and t_4 coloured in green, blue, orange and red, respectively.

during a particular time period, trajectory t_3 is retained as it follows this predefined path. Trajectories t_1, t_2 and t_4 are excluded from consideration since they involve detours. In this particular example, we argue that traffic flow within the orange path is represented better by only considering t_3, although all trajectories touch the path.

Following the generation of time series data, we construct suitable Machine Learning (ML) models to address the specific problem at hand. In this research, we evaluate two such models: an ensemble-based decision tree model using XGBoost [7], and a model based on Long Short-Term Memory (LSTM) [8], an evolution of the Recurrent Neural Network (RNN). We also analyze the advantages and disadvantages of using each model for forecasting traffic flow along predefined paths.

This paper aims to address the problem of effectively predicting traffic flow on road networks. While several studies have been conducted regarding this topic, the novelty of our work stands on that it proposes a distinct path-based methodology that focuses on solving this problem. To the best of our knowledge, no similar research has been published in this field yet.

In practice, the role of the proposed framework is to assist urban traffic management systems in guiding and managing traffic flow, thus controlling traffic volume, avoiding traffic congestion, and constructing an efficient road network.

To summarise, the main contributions of this paper are listed as follows:

- We propose a novel methodology for predicting traffic flow on paths within a road network. Our methodology is generic as it based on timeseries forecasting, and towards this direction, we build an ensemble-based and RNN-based model.
- We demonstrate the efficacy of our proposal by a thorough experimental study using a real-world dataset.

The rest of this paper is organised as follows: Section 2 discusses related work. In Section 3, preliminary terms and definitions are provided. In Section 4, we introduce our methodology. Section 5 presents the experimental study, followed by conclusions in Section 6.

2. Related Work

Several studies approach the problem of predicting traffic flow on road networks as follows: given a dataset of historical traffic flow values, which consists of information about the number of vehicles moving through specific sections of the road network between intersections or multiple areas within the same road network monitored by traffic sensors, the objective is to predict the future traffic flow at these locations at a given time in the future. These studies mainly focus on ML approaches.

Many researchers have tried algorithms like Support Vector Machines (SVM) [9, 10], K-Nearest Neighbors (KNN) [9], Bayesian networks [9], Autoregressive Integrated Moving Average (ARIMA) [10, 11, 12], Shallow Neural Networks (SNNs)[11], Deep Neural Networks (DNNs) [13, 11, 10, 14] and hybrid implementations [9], each with its strengths and weaknesses. For example, SVM and KNN are capable of capturing complex patterns in multidimensional data [9]. SNNs, while useful in many applications, struggle with non-linear patterns of traffic flow [11]. ARIMA models perform well in predicting short-term traffic [11], making them useful for scenarios where precise, immediate forecasts are needed. Last but not least, hybrid approaches leverage the unique strengths of many models combined to achieve better predictions.

Due to the inherent volatility of traffic flow patterns, DNNs become essential [11, 13]. Numerous studies in the literature have recognized this necessity. In a specific instance, researchers introduced a traffic flow prediction framework centred on Fully Connected Neural Networks (FCNN) [11]. This model utilized historical traffic flow data to predict traffic conditions one timestep ahead. Notably, the authors enhanced model performance and generalization by incorporating optimization techniques such as Batch Normalization (BN) and dropout layers.

Another aspect of addressing the traffic flow prediction problem using DNNs involves the utilisation of Stacked Autoencoders (SAEs) [13]. The SAEs were used to distil meaningful information from traffic data by compressing input data into a lower-dimensional representation and reconstructing the original data. Comparative evaluation against alternative ML algorithms, such as SVM, Random Walk (RW), and Radial Basis Function Neural Network (RBFNN), underscored the superior performance of the proposed SAE model, particularly in situations characterised by high traffic flow volumes.

However, the most common deep learning approach proposed when utilising time-related historical data involves the use of an LSTM model. For instance, in [10],

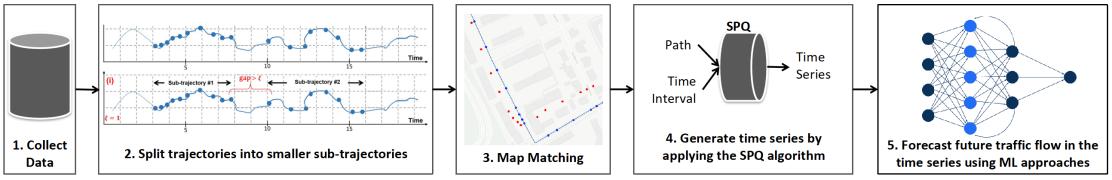

Figure 2: The complete workflow of the proposed methodology.

this model was employed by researchers to take historical traffic flow data as input and predict short-term traffic patterns within a 15-minute time horizon in the future. The LSTM exhibited superior performance compared to SVM and Feedforward Neural Networks (FNN), effectively addressing traffic flow's non-linear and time-dependent characteristics. In another study, scientists introduced the DeepCrowd model [14], which integrates Convolutional LSTM Neural Networks with High-Dimensional Attention Mechanisms (HDAM) to capture spatial and temporal traffic data correlations.

In [15] the authors present the Foresight cloud-based system for real-time spatio-temporal forecasting. The paper discusses practical aspects to implement a real-time system for real-world, while it presents experimental results of the application of several ML methods to the problem at hand. In addition, the authors adapt spatio-temporal ML methods to incorporate dynamic urban events and vehicle-level flow information into the predictive models.

Despite the widespread use of LSTM, SAE, and FCNN models for handling time-related data, another study introduces the power of Graph Neural Networks (GNNs) when dealing with traffic data. The proposed methodology, [12], leverages historical data, incorporating factors influencing traffic flow and speed, including time, space, weather conditions and activities such as public holidays. The chosen model for this approach is the Graph Hierarchical Convolutional Recurrent Neural Network (GHCRNN), which is specifically designed for urban areas. The GHCRNN model integrates both spatial and hierarchical information to improve prediction accuracy. It includes convolutional layers for spatial analysis, pooling units for hierarchical structure representation and an encoder-decoder architecture utilising Gated Recurrent Unit (GRU) modules. The experimentation shows the proficiency of GHCRNN in handling extensive graphs and delivering precise predictions.

Besides the powerfulness of neural networks, the following study proves that ML implementations also address the traffic flow prediction problem with high accuracy. The XGBoost algorithm with the Wavelet Decomposition (WD) technique are used to predict short-term traffic flow [9]. This method aims to extract additional information from the feature to be predicted, combining low and high-frequency details during the reconstruction process. The model is compared with other algorithms, such as the SVM and the XGBoost, without the use of WD technique. Results indicate that the proposed approach achieves superior performance.

In summary, the examined studies showcase notable progress in the domain of traffic flow prediction. However, these researches primarily focus on predicting future traffic flow in specific locations, whether it is a single road segment between intersections or many road segments within intersections, each treated independently. This differs from our objective, which involves forecasting traffic flow within paths. To the best of our knowledge, this study represents a novel contribution to the field.

3. Preliminaries

In this section, we introduce some basic terms required in the rest of the paper and formulate the problem at hand.

In our setting, trajectories are moving on a road network. A road network is defined as a directed graph, denoted as G=(V,E), with V=$\{v_1, v_2, ...v_n\}$ representing the set of vertices in the graph G and E=$\{e_1, e_2, ...e_m\}$ representing the set of edges in the graph G. In a general context, when considering a graph that represents a road network, set V is understood to denote the intersections within the network, while set E signifies the road segments between two intersections. Within a road network represented as a graph G, a path, also a pathway, is defined to be a sequence of at least two consecutive edges in G.

In the context of a road network, diverse moving objects (MO) navigate through it. An MO is defined as any entity that travels along the edges of a road network represented as a graph G. Examples of MOs include cars, buses and taxis.

The route of a MO is defined as a sequence $<r_1, r_2, ..., r_n>$, where each element r_i is a triple (t_i, lon_i, lat_i) denoting the timestamp of the MO's sampling and the respective GPS coordinates.

Based on the provided definitions, the problem addressed in this study is formulated as follows: given (i) a graph G representing a road network, (ii) a dataset D containing a set of routes of MOs traveling along the road network represented by G and (iii) a set P of predefined

paths within G, the goal of traffic flow prediction is to predict the occupancy of each path in P, i.e. the number of the MOs that will traverse each predefined path within a specific time horizon in the future using D as its knowledge base.

4. Methodology

In this section, we introduce the proposed methodology for efficiently resolving the challenge of forecasting traffic flow along a pathway within a road network. The workflow of the proposed methodology is depicted in Figure 2. The methodology starts by processing a collection of GPS data that contain routes of MOs, and more specifically by performing a route-splitting (i.e., segmentation) operation. Route splitting is the process of partitioning the entire route of a specific MO into distinct trajectories, guided by specific conditions. Subsequently, the resulting trajectories undergo map matching onto the road network. The next step in the methodology is to measure historical traffic flow volumes along each path that belongs in the set of predefined paths and at uniform-sized time intervals, resulting in the creation of a time series dataset for each predefined path. After generating the time series data, we extract temporal features from timestamps to enhance the prediction accuracy of the forecasting models. Finally, prediction procedures of traffic flow magnitude are conducted using two ML models, based on the XGBoost and the LSTM RNN approaches.

4.1. Dataset Preprocessing

In the initial phase, it is crucial to acknowledge the potential existence of routes of MOs that contain elements that happen to be recorded at sparse timestamps. To address this problem, a necessary step involves breaking down MO routes into trajectories. The time gap between any two consecutive elements within the same trajectory is constrained to a specific time value, denoted as Dt. A route undergoes a split when consecutive pairs of elements within the route exhibit a time difference greater than the specified value Dt.

The subsequent phase involves the map-matching process. The spatial locations of the trajectories may not precisely align with the road network due to potential noise during sampling or technical errors. Hence, the usage of a map-matching algorithm becomes imperative to ensure accurate alignment with the road network.

For precise map-matching of trajectory coordinates (*lat* and *lon*) onto the road network, the Valhalla Meili API[2], recognized for its highly efficient map-matching algorithm [16], is utilized.

[2]https://valhalla.github.io/valhalla/api/map-matching/api-reference/

Once the split trajectories are processed through the map-matching algorithm, the information given as output encompasses the edges of the road network that the trajectory traversed in chronological order. Additionally, it includes information about the time interval the trajectory lied on each edge.

4.2. Generating Traffic Flow Time Series

The subsequent stage in the proposed methodology involves the generation of time series data using the trajectory data. Time series data represent the traffic flow inside each pathway under study at equal-sized time intervals. The utilisation of the Strict Path Queries (SPQs) algorithm [6] is crucial for this task. This type of query identifies all trajectories strictly following a specified path within a defined time interval. The start of this interval signifies the minimum acceptable timestamp for the trajectory to have entered the designated path, while the end of the interval indicates the tolerated timestamp for the trajectory to have exited it.

The first parameter that the SPQ algorithm requires is a path in which past traffic flow values will be measured at uniform-sized time intervals. The second parameter essential for the SPQ algorithm is a time interval. Because the SPQ algorithm is employed to generate time series data, uniform time intervals must be established. To achieve this, the total sampling duration of trajectories within dataset D is divided by a positive number, represented as Q. This division results in the creation of smaller time sub-intervals, each characterized by a duration of Q time units.

For each sub-interval, the SPQ algorithm measures traffic flow along the predefined path, producing a time series with traffic flow volume per sub-interval.

To improve the quality of time series data, we also extract time-related features from the timestamps. These features encompass the day (numerical representation, e.g., 1, 2, 3), the day of the week (e.g., Monday, Tuesday), the hour (numerical representation from 0 to 23), and the minutes (numerical representation from 0 to 59). To account for the cyclical nature of time, cyclic encoding is applied to these time-related features, capturing their periodic characteristics using semitones and cosines.

Incorporating additional time-related information, we introduce the *3-hour-interval* attribute (numerical representation from 1 to 8) to signify the specific 3-hour interval of the day to which a particular recording within the time series belongs.

4.3. Forecasting Models

Resulting time series data serve as the input for the two ML models, XGBoost [7] and LSTM RNN [8], we evaluate in this study. These models are specifically trained to

predict traffic flow on each predefined path. This task is categorized as a supervised ML process.

To structure the data into input and output sets, the sliding window technique is applied. This technique involves dividing the time series into subsets using a moving or sliding window. As the window progresses step by step through the time series, it captures fixed-size segments of historical values. The length of this window determines the amount of past data used to forecast the subsequent value in the time series and corresponds to parameter *past-observations* that actually indicates the count of historical observations utilized for prediction.

Figure 3 illustrates the operation of the sliding window technique. The table at the left displays samples of traffic flow on a single path at 4 uniform-sized time intervals t_1, t_2, t_3 and t_4. For instance, equal to 4 at time interval t_1, 5 at t_2, and so on. If the decision is made to input traffic flow values from the two preceding time steps into the forecasting model to predict the next value, the window length is set to 2. Adhering to these criteria, the time series undergoes a transformation into the format showcased in the table at the right.

Time Interval	Traffic Flow (Count)
t_1	4
t_2	5
t_3	8
t_4	6

Traffic Flow (t-2)	Traffic Flow (t-1)	Traffic Flow (t)
4	5	8
5	8	6

Figure 3: The sliding window technique applied on time series data (left: samples of traffic flow on a path; right: transformation according to a window length equal to 2).

Through the implementation of the sliding window technique, we can generate input and output sets for different values of the *past-observations* parameter. These sets are utilized for training and evaluating both machine learning models. Each model receives an input vector with L features, including historical traffic flow values, temporal characteristics with their cyclic encodings, and the *3-hour-interval*. Furthermore, the models incorporate the mean and variance of historical traffic flow values as input. The goal of each model is to predict the traffic flow in the next sub-interval with duration Q.

To improve efficiency, a correlation matrix is utilized. This matrix helps identify and remove attributes that redundantly convey similar information, ensuring only those with low correlation are included. Features with low correlation are chosen for the training and testing phases of XGBoost and LSTM models.

In Figure 4, the architecture of the selected XGBoost model is illustrated. Using a set of 100 XGBoost decision

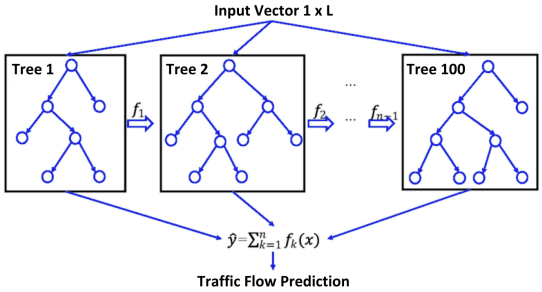

Figure 4: Architecture of XGBoost model.

trees, the model takes as input a vector with L features in order to predict future traffic flow. Each subsequent tree is designed to improve upon its predecessor. The ultimate prediction for a regression task is derived by consolidating the predictions made by each individual tree within the ensemble. This aggregation process ensures that the final forecast benefits from the collective contributions of all the trees in the model.

On the other hand, Figure 5 depicts the proposed LSTM model. The neural network is designed to take a vector with L features as input. This vector is then reshaped in a 3-dimensional tensor. The information traverses through 3 LSTM layers with 50, 25, and 12 units, respectively. Dropout layers are strategically employed right after the LSTM layers to deactivate certain neurons and prevent overfitting during the training process. Subsequently, the information passes through 3 fully connected layers with 6, 3, and 1 neuron, respectively, before producing the final output y. ReLu activation functions are utilized in all LSTM layers, while the fully connected layers employ a linear activation function.

5. Experimental Study

In this section, we evaluate the proposed methodology using a real-world trajectory dataset. The source code that implements the proposed methodology is available at GitHub[3].

In particular, we used the Cabspoting dataset[4], a popular urban mobility dataset consisting of navigation routes from 537 taxis in San Francisco, CA, USA, recorded from May 17, 2008, to June 10, 2008. The routes are captured at an average sampling rate of approximately 30 seconds. The entire dataset encompasses 11,220,491 GPS records.

The research was conducted using a combination of computational resources. Data preparation and time series generation procedures were performed on a Dell Inspiron 15 3000 series laptop equipped with an Intel Core i5-4210U processor and 8GB of RAM. For model

[3]https://github.com/stratoskar/Path_Based_Traffic_Flow_Prediction
[4]https://ieee-dataport.org/open-access/crawdad-epflmobility; https://stamen.com/work/cabspotting/

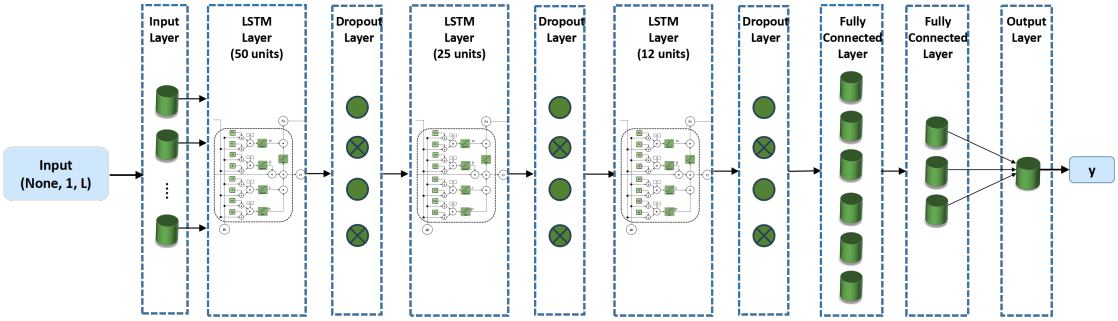

Figure 5: Architecture of the LSTM RNN model used.

training and forecasting processes an NVIDIA Tesla T4 GPU hosted on Google Colab was utilized. This GPU provided the necessary computational power to train the ML models efficiently.

Following the proposed methodology, the segmentation of taxi routes into trajectories was performed using a time threshold of Dt=90 seconds. The identification of the route of a taxi before splitting is encoded as *taxi-id*, while the subsequent identification of each newly generated trajectory is denoted as *traj-id*. Thus, each trajectory in the dataset is uniquely identified by a pair of *taxi-id* and *traj-id* values. The produced trajectory dataset encompasses a total of 521,135 unique trajectories.

The Valhalla API receives GPS coordinates for each element in a trajectory as input and subsequently produces a potential map-matched trajectory that aligns with the road network of San Francisco. Following this procedure, the resulting dataset includes details, such as the unique trajectory identifier (denoted by *taxi-id* and *traj-id*) and the exact sequence of road network edges traversed by the trajectory. The updated dataset from the Valhalla Meili map-matching algorithm also provides timestamps denoted as *t_enter*, indicating the moments when the trajectory enters the specified edges.

The next step involves constructing a time series dataset focusing on the recording of traffic flow by path and time interval. The available map-matched trajectory data spans from May 17th, starting at 10:00:04, to June 10th, concluding at 09:00:04. This time frame is segmented into smaller intervals of half an hour each, i.e. Q = 30 min.

To quantify traffic flow along pathways, we choose to create a set of 100 unique and distinct paths. The length of these paths, determined by the number of edges, spans from 2 to 20 edges. Importantly, these paths are not randomly generated; rather, they are derived from the trajectory data. This ensures that each path created is a subset of one or more trajectories.

The construction of a time series utilizes the SPQ algorithm. For each call of the SPQ algorithm, the input consists of the specific path for which traffic flow needs to be counted, along with the corresponding half-hour time interval. By repeating this procedure for each path and every interval, we create a traffic flow time series for each path.

The application of the SPQ algorithm filters out the trajectories that deviate while moving along a specific path. Figure 6 illustrates the difference in traffic flow data obtained when using the SPQ algorithm on the set of 100 predefined paths and data collected without the use of SPQ on the same set of predefined paths. In the absence of SPQ restrictions, traffic flow on each route is calculated by counting the trajectories that have traversed all path edges at least once within a defined time interval. However, when SPQ restrictions are applied, it is evident that the collected traffic flow data are reduced.

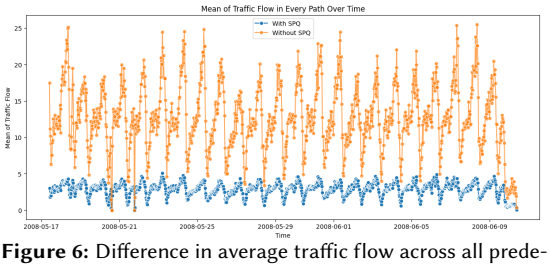

Figure 6: Difference in average traffic flow across all predefined paths grouped by timestamps: dataset 1 (blue) is formed using SPQ algorithm, whereas dataset 2 (orange) is developed without adhering to SPQ rules.

To build robust and highly precise ML models, it is crucial to selectively choose attributes from the time series data with low correlation. To accomplish this, a correlation matrix is utilized. The correlation matrix presented in Figure 7 indicates that the temporal characteristics *day*, *dayofweek*, *hour* and *minute* share common information with their corresponding cyclic encoded features *day_of_week_sin*, *day_of_week_cos*, *hour_sin*, *hour_cos*, *minute_sin*, *minute_cos*, *day_sin* and *day_cos*. As a result, temporal attributes *day*, *dayofweek*, *hour*, and *minute* are eliminated from the time series data. The rest of the attributes present in the heatmap representation are taken into account in the training and evaluation procedures of the ML models.

Time series data is divided into training and testing

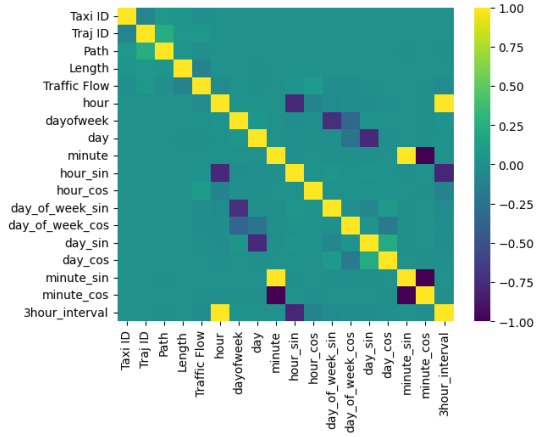

Figure 7: Heatmap illustrating the correlation matrix among features in the time series data.

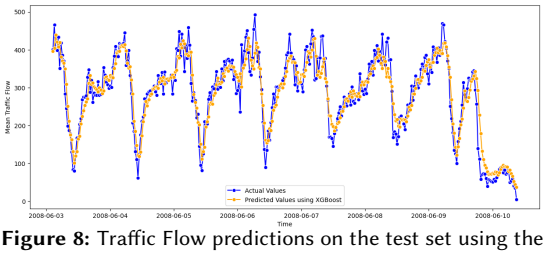

Figure 8: Traffic Flow predictions on the test set using the XGBoost model.

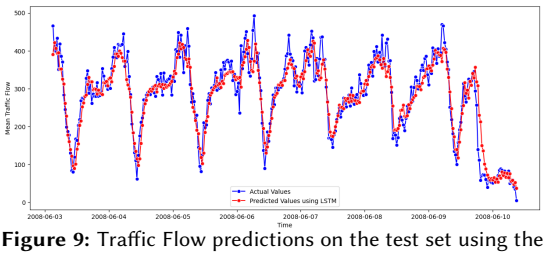

Figure 9: Traffic Flow predictions on the test set using the LSTM model.

sets, with the training set encompassing the earliest data in the time series, spanning from May 17th to June 2nd inclusive. The remaining data forms the testing set.

The subsequent phase involves fine-tuning XGBoost's hyperparameters utilizing the Grid Search Cross-Validation method[5]. The hyperparameters targeted for optimization encompass regularization parameters, such as gamma, alpha, lambda, the learning rate influencing the backpropagation algorithm, as well as the maximum depth of each decision tree. Employing the best combination of hyperparameter values, we train the XGBoost model utilizing an ensemble of 100 trees. The test set is used to evaluate the performance of XGBoost in terms of RMSE and MAE regression evaluation metrics.

Concerning the LSTM model, the time series data within the training set undergo a reshaping process, resulting in a 3-dimensional tensor. The output y of the LSTM model represents the predicted traffic flow value in the time series. The LSTM model is then evaluated on the test set using the same evaluation metrics, RMSE and MAE.

To determine the optimal value for the *past-observations* parameter, we employ varying sliding window lengths. Both models are assessed on the same test set for different *past-observations* values, specifically 2, 3, 4, 5, and 6. The outcomes are presented in Table 1 using the data created using the SPQ algorithm. Table 2 indicates the result of the same experiment using the data collected when the usage of SPQ is absent. The comparison of the two tables makes clear that by applying the proposed methodology we succeed lower RMSE and MAE scores.

Table 1
Performance comparison of XGBoost and LSTM models using the SPQ data.

	XGBoost		LSTM	
past-observations	RMSE	MAE	RMSE	MAE
2	2.3245	1.3210	2.3886	1.4630
3	2.3081	1.3052	2.3000	1.3556
4	2.2961	1.2929	2.3009	1.3738
5	**2.2821**	**1.2870**	2.3002	1.3189
6	2.2890	1.2879	**2.2643**	**1.2510**

Table 2
Performance comparison of XGBoost and LSTM models using data created without applying the SPQ algorithm.

	XGBoost		LSTM	
past-observations	RMSE	MAE	RMSE	MAE
2	11.2448	5.5897	11.3756	5.9963
3	11.1362	5.4913	11.4908	6.0612
4	11.1243	5.4591	11.2740	5.7380
5	11.1279	5.4485	**11.2721**	5.6650
6	**11.0943**	**5.4437**	11.3616	**5.6101**

Forecasts are produced using the known values from the test set. Based on the conclusions drawn from Table 1, the XGBoost model is configured with a setting of 5 (6, respectively) for the *past-observations* parameter. Figures 8 and 9 showcase the predictions made by both models on the test set. The blue colour represents the observed sum of traffic flow values in the test set, aggregated based on timestamps. Conversely, the orange and red colours represent the predicted sum of traffic flow values across all 100 predefined paths in the test set, organized by

[5]https://scikit-learn.org/stable/modules/grid_search.html

timestamp and generated using the trained XGBoost and LSTM models, respectively. It appears that both models effectively capture the trend and seasonality present in the time series data of the test set.

6. Conclusion

In this study, we presented a generic methodology for forecasting traffic flow volumes on pathways, employing ML algorithms, with a particular emphasis on the XGBoost and the LSTM models. When tested on a real-world dataset containing taxi routes in the San Francisco area, both models demonstrated good performance, which is evidently better than the case when all data is used.

A limitation that constrained the scope of our study is the reliance only on taxi traffic data resulted in an incomplete representation of the city's overall traffic dynamics, overlooking the movements of other transportation modes, such as buses and private cars. Furthermore, a more holistic comprehension of traffic flow prediction dynamics could be achieved by integrating additional data, such as vehicle accidents and weather conditions, which are known to impact traffic flow patterns.

Lastly, for future research endeavours, it is crucial to simulate scenarios using diverse modelling approaches, including Graph Neural Networks (GNNs), to assess their effectiveness in capturing complex relationships within the road network. This comprehensive approach aims to contribute to a more nuanced and adaptable predictive framework for future traffic flow predictions.

Acknowledgments

This work was supported by the Horizon Europe research and innovation programmes under GA 101070416 (Green.Dat.AI) & 101093051 (EMERALDS) funded by EU.

References

[1] Y. Xu, L. E. Olmos, D. Mateo, A. Hernando, X. Yang, M. C. González, Urban dynamics through the lens of human mobility, Nature Computational Science 3 (2023) 611–620.

[2] N. Pelekis, Y. Theodoridis, Mobility Data Management and Exploration, Springer New York, 2014.

[3] G. Vouros, G. Andrienko, C. Doulkeridis, N. Pelekis, A. Artikis, A.-L. Jousselme, C. Ray, J. Cordero, Big Data Analytics for Time-Critical Mobility Forecasting, Springer Nature, 2020.

[4] C. Doulkeridis, A. Vlachou, N. Pelekis, Y. Theodoridis, A survey on big data processing frameworks for mobility analytics, SIGMOD Rec. 50 (2021) 18–29.

[5] E. Chondrodima, N. Pelekis, A. Pikrakis, Y. Theodoridis, An efficient lstm neural network-based framework for vessel location forecasting, IEEE Transactions on Intelligent Transportation Systems 24 (2023) 4872–4888.

[6] B. Krogh, N. Pelekis, Y. Theodoridis, K. Torp, Path-based queries on trajectory data, in: Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL '14, Association for Computing Machinery, New York, NY, US, 2014, pp. 341–350.

[7] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, 2016.

[8] R. C. Staudemeyer, E. R. Morris, Understanding lstm – a tutorial into long short-term memory recurrent neural networks, 2019. arXiv:1909.09586.

[9] X. Dong, T. Lei, S. Jin, Z. Hou, Short-term traffic flow prediction based on xgboost, in: 2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS), 2018, pp. 854–859.

[10] Y. Tian, L. Pan, Predicting short-term traffic flow by long short-term memory recurrent neural network, in: 2015 IEEE International Conference on Smart City/SocialCom/SustainCom, 2015, pp. 153–158.

[11] H. Tampubolon, P.-A. Hsiung, Supervised deep learning based for traffic flow prediction, in: 2018 International Conference on Smart Green Technology in Electrical and Information Systems (ICSGTEIS), 2018, pp. 95–100.

[12] M. Lu, K. Zhang, H. Liu, N. Xiong, Graph hierarchical convolutional recurrent neural network (ghcrnn) for vehicle condition prediction, 2019. arXiv:1903.06261.

[13] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang, Traffic flow prediction with big data: A deep learning approach, IEEE Transactions on Intelligent Transportation Systems 16 (2015) 865–873.

[14] R. Jiang, Z. Cai, Z. Wang, C. Yang, Z. Fan, Q. Chen, K. Tsubouchi, X. Song, R. Shibasaki, Deepcrowd: A deep model for large-scale citywide crowd density and flow prediction, IEEE Transactions on Knowledge and Data Engineering 35 (2023) 276–290.

[15] C. Conlan, J. Oakley, G. V. Demirci, A. Sfyridis, H. Ferhatosmanoglu, Real-time spatio-temporal forecasting with dynamic urban event and vehicle-level flow information, in: CEUR Workshop Proceedings, volume 3379, 2023.

[16] S. Saki, T. Hagen, A practical guide to an open-source map-matching approach for big gps data, SN Computer Science 3 (2022).