International Journal on Artificial Intelligence Tools
Vol. 32, No. 1 (2023) 2350008 (27 pages)

DOI: 10.1142/S0218213023500082

World Scientific
www.worldscientific.com

Novel Sparse Feature Regression Method for Traffic Forecasting

Athanasios I. Salamanis*, George A. Gravvanis†

Department of Electrical and Computer Engineering
School of Engineering, Democritus University of Thrace
GR-67100 Xanthi, Greece
**asalaman@ee.duth.gr*
†*ggravvan@ee.duth.gr*

Sotiris B. Kotsiantis‡, Michael N. Vrahatis§

Department of Mathematics, University of Patras
GR-26110 Patras, Greece
‡*sotos@math.upatras.gr*
§*vrahatis@math.upatras.gr*

Received 27 May 2022
Accepted 14 August 2022
Published 28 February 2023

Traffic forecasting is an integral part of modern intelligent transportation systems. Although many techniques have been proposed in the literature to address the problem, most of them focus almost exclusively on forecasting accuracy and ignore other important aspects of the problem. In the paper at hand, a new method for both accurate and fast large-scale traffic forecasting, named "sparse feature regression", is presented. Initially, a set of carefully selected features is extracted from the available traffic data. Then, some of the initial features are sparsified, namely they are transformed into sets of sparse features. Finally, a linear regression model is designed using the sparse feature set, which is trained by solving an optimization problem using a sparse approximate pseudoinverse as a preconditioner. We evaluated the proposed method by conducting experiments on two real-world traffic datasets, and the experimental results showed that the method presents the best balance between accuracy of predictions and time required for achieving them, in comparison with a set of benchmark models.

Keywords: Traffic forecasting; feature engineering; feature sparsification; linear regression; sparse least squares; computational efficiency.

†Corresponding author

Abbreviations

AM	Autoregressive Model
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
ARR	Adjusted Ratio of Ratios
BPNN	Back Propagation Neural Network
DCRNN	Diffusion Convolutional Recurrent Neural Network
DFS	Dense Feature Set
DL	Deep Learning
EPCGLS	Explicit Preconditioned Conjugate Gradient Least Squares
FA	Forecasting Accuracy
FNN	Feedforward Neural Network
GNN	Graph Neural Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HA	Historic Average
HM	Heteroscedasticity Models
ITS	Intelligent Transportation Systems
KF	Kalman Filtering
kNN	k-Nearest Neighbors
LSTM	Long Short-Term Memory
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
PeMS	Performance Measurement System
RBF	Radial Basis Function
RMSE	Root-Mean-Square Error
RNN	Recurrent Neural Network
SFR	Sparse Feature Regression
SFS	Sparse Feature Set
SSA	Singular Spectrum Analysis
STARIMA	Space-Time Autoregressive Integrated Moving Average
STGCN	Spatio-Temporal Graph Convolutional Network
SVR	Support Vector Regression
VAR	Vector Autoregressive Model
VDS	Vehicle Detection Stations

1. Introduction

Modern *Intelligent Transportation Systems* (*ITS*) rely on accurate and timely traffic predictions in order to provide services like traffic management, automatic incident detection,[1] congestion avoidance and vehicle routing.[2] Hence, it is very important for an ITS to have a subsystem that will be able to accurately and timely predict traffic for an entire network.

Traffic forecasting is a complex task given the spatiotemporal dependencies of the elements of the traffic network, the highly volatile nature of the traffic variables, and the scale of traffic networks. Given its importance and complexity, traffic forecasting has attracted the interest of many researchers in the last decade. Many different approaches for traffic forecasting have been reported in the literature, based on *statistics*, *Machine Learning* (*ML*) and, quite recently, *Deep Learning* (*DL*). At the present time, the models that are considered as state-of-the-art in terms of forecasting accuracy are the global spatiotemporal deep learning models.[3,4] In particular, this is a single deep learning model designed for the whole traffic network that contains a block for extracting the spatial dependencies between the elements of the network, another block for extracting the temporal dependencies within the traffic data, and a block for integrating the spatiotemporal features extracted by the previous blocks in order to produce predictions. Due to their complexity and nonlinearity, such models have produced very accurate predictions in several different settings and forecasting scenarios.[5,6]

Despite the high accuracy of their predictions, the global spatiotemporal deep learning models have some inherent disadvantages. In particular, they require large amounts of data and significant time to be trained, their forecasting accuracy is very much dependent on the appropriate values of their hyperparameters and therefore they require a lot of time for hyperparameter tuning, and they are *black box* models, namely their results cannot be easily explained to decision makers. In addition, due to their global unified nature (i.e., one model for the whole traffic network), their training process is difficult to be parallelized. This means that for a prediction to be available for a specific element in the network, the whole large model should be trained first. Even with the current advances in GPU-accelerated training, this process might require a considerable amount of time. This is an important problem mainly in real-time traffic forecasting scenarios in which the models need to be retrained regularly in order to adapt to the ever-changing network traffic. Hence, it becomes evident that even though the current state-of-the-art traffic forecasting models can achieve high forecasting accuracy, they neglect the real-world constraints of a traffic forecasting system, like the availability of data and computational resources.

Given the limitations of the deep learning models, our goal is to design simpler and faster models for accurate traffic forecasting. In this context, we propose a simple, fast and effective regression model for large-scale traffic forecasting. The model integrates spatiotemporal correlations between traffic elements by introducing a new set of carefully selected features. The initial feature set is then transformed into a sparse feature set, and a linear regression model is designed using its features. The parameters of the model are estimated through the solution of an optimization problem by a preconditioned iterative method. The main contribution of the paper at hand is the introduction of a simple yet effective and fast large-scale traffic prediction model that exhibits both good forecasting accuracy and reduced computational complexity.

The rest of the paper is organized as follows. In Sec. 2, we give a short presentation of other traffic forecasting approaches from the literature. In Sec. 3, we provide a detailed description of the proposed method, while in Sec. 4 we describe the datasets used in this work as well as the experimental setup. In Sec. 5, we demonstrate the experimental results, and in Sec. 6 we present a synopsis and concluding remarks of our work along with some future research directions.

2. Related Work

Due to its value for the successful deployment of *Intelligent Transportation Systems* (*ITS*), traffic forecasting has attracted the interest of many researchers in the last decade. The various models reported in the literature can be grouped in two main categories, namely the *parametric* and the *nonparametric* models. A parametric model has a predefined structure and only its parameters are derived from the available data, while in a nonparametric model all key elements are learned from the data. Moreover, there is another intermediate category that includes *hybrid* models that attempt to combine characteristics from both categories in order to produce better predictions.

Some of the most popular traffic forecasting models belong to the parametric category. These models are based on the typical *Autoregressive Model* (*AM*) and its several variants, e.g., *Autoregressive Integrated Moving Average* (*ARIMA*), *Spatiotemporal ARIMA* (*STARIMA*), etc.[7–10] as well as the *Kalman Filtering* (*KF*) techniques[11–13] and the *Heteroscedasticity Models* (*HM*).[14–16] For example, Schimbinschi *et al.*[17] proposed a generic topology-regularized *Vector Autoregressive Model* (*VAR*), which leverages the spatiotemporal dependencies between network elements in order to achieve higher forecasting accuracy compared to benchmarks. In addition, Zhou *et al.*[18] presented a KF technique that exploits the discrepancy between the predictions of a traditional KF and a random walk model in order to achieve high forecasting accuracy. Moreover, Pavlyuk[19] highlighted the importance of AMs in the transportation domain.

Our method has certain similarities with the models of this category. In particular, it is a linear model as many of the models of this category and it uses features manually extracted from the data, which is the most common feature extraction approach for the parametric models. However, an important difference of the proposed model with the models of this category is that it combines continuous with categorical features, when most of the parametric models use only continuous features (i.e., the lagged values from the traffic time series).

On the other hand, most nonparametric approaches for traffic forecasting are based on machine learning regression models, namely *Artificial Neural Networks* (*ANNs*), *Support Vector Regression* (*SVR*) and *k-Nearest Neighbors* (*k*NN). For example, Cheng *et al.*[20] introduced an adaptive *k*NN model that captures the spatiotemporal dependencies between road segments and exploits them in order to produce accurate traffic predictions. Additionally, Sun *et al.*[21] proposed a fully automatic dynamic procedure that enables self-adjustment of the parameters of a

kNN-based traffic forecasting model. Moreover, Liu *et al.*[22] investigated the combination of a kNN and a SVR model in order to perform short-term traffic flow forecasting, while Cai *et al.*[23] built a kNN regression model that re-balances the traffic time series by rejecting those identified as outliers. Another interesting approach is described in the work of Hong *et al.*,[24] in which the authors introduced a multi-channel feature extraction process along with an updated metric for identifying neighbors in order to enhance the forecasting capacity of a kNN regression model. Furthermore, Kolidakis *et al.*[25] studied the combination of a shallow ANN architecture with *Singular Spectrum Analysis* (*SSA*) and concluded that this fusion can result into better predictions compared to conventional shallow ANN architectures.

The new traffic forecasting method has many differences compared to models in this category. In particular, it has a predefined linear structure through which it generates its predictions, while the nonparametric models have no predefined structure and most of them are nonlinear. In addition, while in both SVMs and ANNs the model parameters are estimated by solving an optimization problem, the approaches used for solving these problems differ from that of the proposed method. In particular, in SVMs and ANNs the underlying optimization problems are usually solved using the method of Lagrange multipliers and the gradient descent method, respectively, while in the case of the proposed model the optimization problem is solved using a sparse approximate pseudoinverse.[26] In addition, while the nonparametric models use handcrafted features that may contain both continuous and categorical ones, they do not integrate a mechanism for sparsifying the categorical features as the proposed model does. Moreover, while in both kNN and the proposed method the effect of neighboring elements in taken into account in the modelling process, in the former case the neighbors are identified by computing the distances between feature vectors, while in the latter they are identified through the computation of spatial distances.

The nonparametric models have low capacity in terms of the number of parameters, and hence they cannot effectively capture the complex spatiotemporal dependencies existing in large-scale traffic *Deep Learning* (*DL*) networks. To address this issue, global spatiotemporal *Deep Learning* (*DL*) models have been suggested by several researchers, which mainly consist of one block for capturing the spatial dependencies of traffic network elements, one for identifying the temporal dependencies and one for combining the features extracted by the other two blocks.[27] Nowadays, in most cases, the block that captures the spatial dependencies in the traffic network is a *Graph Neural Network* (*GNN*),[28] while the block that identifies the temporal dependencies is either a type of a *Recurrent Neural Network* (*RNN*), like the *Long Short-Term Memory Network* (*LSTM*)[29,30] or the *Gated Recurrent Unit* (*GRU*),[31] or a block that implements the *attention mechanism*.[32] For example, Yu *et al.*[3] introduced convolution operations on graphs using first-order Chebyshev polynomials[33] in order to capture the spatial dependencies of the elements of the network, and gated convolutions on the traffic time series to

extract the temporal dependencies. The resulting *Spatio-Temporal Graph Convolutional Network* (*STGCN*) achieved better forecasting accuracy compared to several benchmark models in two different datasets. Several variants of this approach have been proposed in the literature.[34–38]

From a different perspective, Li *et al.*[5] proposed the *Diffusion Convolutional Recurrent Neural Network* (*DCRNN*), namely a DL framework in which the traffic flow is modelled as a diffusion process on a directed graph resulting in representing the spatial dependencies using bidirectional random walks, while the temporal dependencies are represented using an encoder-decoder architecture with scheduled sampling. The authors reported a 12%–15% improvement in forecasting accuracy compared to both parametric and nonparametric benchmark models. Additionally, Mallick *et al.*[4] presented an approach that overcomes the computational and memory bottlenecks of the DCRNN model by decomposing a large highway network into smaller networks and training separate DCRNN models independently. The same rationale was followed by other researchers as well.[6,39,40] Furthermore, researchers proposed DL architectures in which the temporal dependencies are captured by a block implementing the attention mechanism.[41–45]

Our approach differs in many respects from the deep learning models. In particular, the proposed method is local in the sense that a separate instance of the model is built for each traffic element, while most deep learning models used for traffic forecasting are global models. Also, the proposed model is linear while the deep learning models are nonlinear. In addition, the deep learning models automatically identify and extract features from the available traffic data, while the proposed model uses manually extracted features. Both the proposed and the deep learning models are trained by solving an optimization problem, but the deep learning models use a variant of the gradient descent method for solving this problem while the proposed method uses a sparse approximate pseudoinverse.[26] Moreover, the proposed method leverages the sparsification of the categorical features in order to increase its forecasting accuracy and training speed, while most deep learning models do not take any advantage through sparsification. Furthermore, the deep learning models require large amounts of data, significant time to be trained, and their accuracy depends a lot on the values of their large set of hyperparameters. On the other hand, the proposed method does not require much data to yield acceptable accuracy, it is quickly trained, and its unique hyperparameter can be easily and quickly tuned. It should also be noted that while deep learning models are considered *black box* models and hence not explainable, the predictions of the proposed model can be easily explained by the examination of the values of its parameters.

3. Proposed Method

3.1. *Feature selection & construction*

In the paper at hand, a new method for traffic prediction, named *Sparse Feature Regression* (*SFR*), is presented. We denote by $x_{t,d}$ the values of a time series of a

day d with size $n \in \mathbb{N}$ (i.e., $1 \leqslant t \leqslant n$). We also denote by $p \in \mathbb{N}$ the length of a sliding window used for selecting past lagged values, $h \in \mathbb{N}$ the forecasting horizon and $N \in \mathbb{N}$ the number of available traffic time series. The modelling process starts by selecting the initial features of our model. The first part of the initial feature set contains p lagged values, namely:

$$\{x_{t-(p-1),d}, \ldots, x_{t-0,d}\}, \quad p \leqslant t \leqslant (n-h). \tag{1}$$

These are usually the only features used by most parametric traffic forecasting models. Then, we construct the second part of the initial feature set by selecting as feature the temporal index of current traffic time series, namely:

$$\{T\}, \quad T \in \mathbb{N}, \quad T = \{1, 2, \ldots, N\}. \tag{2}$$

Moreover, we select as an additional feature of this part the temporal index of current interval within the current traffic time series, namely:

$$\{t\}, \quad t \in \mathbb{N}, \quad t = \{p, \ldots, n-h\}. \tag{3}$$

We selected these two temporal indexes as features for two reasons. On one hand, we wanted to build a model that will try to learn its exact position in time at each step and try to combine this information with the provided lagged values. The explicit provision of these identifiers of position in time is important because the model cannot possibly infer it just from the lagged values. On the other hand, these features are categorical variables, which means that they can be transformed into sets of binary (sparse) features, thus accelerating the training process of the model by injecting zeros in its coefficient matrix. We refer to these features hereafter as *temporal features*.

In addition to the temporal features, we introduce the concept of spatiotemporal correlation between traffic elements in our initial feature set. For this reason, for each traffic element, we estimated a set of neighboring elements based on spatial distances. In particular, given the fact that (as will be presented in Sec. 4) the traffic elements considered in this work are points uniquely identified by a pair of coordinates, we computed their pairwise geodesic distances, and for each element we kept as neighbors the N_{nbr} nearest elements (i.e., the N_{nbr} elements with the smallest distances). We aggregate the contributions of all neighbors into a single (average) value as follows:

$$x_{t,d}^{(\text{nbr, av})} = \frac{1}{N_{\text{nbr}}} \sum_{j=1}^{N_{\text{nbr}}} x_{t,d}^{(j)}, \tag{4}$$

where $x_{t,d}^{(j)}$ represents the traffic at index t of the time series that corresponds to day d and neighbor j. Based on the above, we defined the last part of the initial feature set as the p lagged values of the time series of the neighboring elements averaged across the neighbors as follows:

$$\{x_{t-(p-1),d}^{(\text{nbr, av})}, \ldots, x_{t-0,d}^{(\text{nbr, av})}\}. \tag{5}$$

In order to accelerate the process of identifying the N_{nbr} neighbors of each element of interest, we used a simple clustering-based approach. In particular, we initially divided the elements of the traffic network into groups using the k-means clustering algorithm, and for each element we calculated its distances only from the elements of its own group and kept the elements with the N_{nbr} smallest distances. This approach significantly shrinks the search space for each element of the traffic network and hence the overall computation time, compared to the brute-force approach of computing for each element of interest its distances with every other element of interest in the traffic network. The brute-force approach has time complexity $O(M^2)$ while our clustering-based approach $O(MN_c)$, where M is the number of elements in the traffic network and N_c the average number of elements per group. With appropriate selection of the number of groups G, it holds that $N_c \ll M$ and hence the time complexity of the clustering-based approach is quite better compared to that of the brute-force approach. A subtle point of the clustering-based approach is that the number of groups should be selected in an appropriate way so that the number of points in each cluster to be greater than the number of requested neighbors N_{nbr}. We used as heuristic for satisfying this condition the equation $G = \sqrt{M/2}$. It is of course obvious that this heuristic cannot guarantee the satisfaction of the condition in all cases.

Based on the above description, the initial *Dense Feature Set* (*DFS*) is formulated as follows:

$$\text{DFS} = \left\{T, t, x_{t-(p-1),d}, \ldots, x_{t-0,d}, x^{(\text{nbr, av})}_{t-(p-1),d}, \ldots, x^{(\text{nbr, av})}_{t-0,d}\right\}. \tag{6}$$

The specific order of the features presented in Eq. (6) facilitates the implementation of our model. It should be noted that no intercept is included as feature in DFS, given the sparsification scheme of the temporal features that will be explained in what follows. The initial DFS contains $D = 2 + 2p$ features in total.

Once the initial dense feature set has been created, the next step is its sparsification, namely the sparsification of the temporal features. In particular, both temporal features are transformed into sets of binary (sparse) features using the dummy encoding scheme. The dummy encoding scheme is similar to one-hot encoding scheme with the key difference that for a categorical variable with K categories the former produces $K-1$ binary variables while the latter K variables. This means that one value of the categorical variable (usually the first or the last one) is encoded by a vector of $K-1$ zeros. Hence in our case, $N-1$ and $n-h-p$ binary features are generated from the temporal features T and t, respectively. For example, for $N = 5$ the value $T = 2$ will be dummy encoded as "0100" and the value $T = 5$ as "0000". The dummy encoding scheme can introduce perfect multicollinearity between the independent variables of a regression model named *dummy variable trap*.[46] We avoid the dummy variable trap by removing the intercept term from our model (as mentioned above). Given this sparsification scheme, the two initial

temporal features are replaced by $(N-1)+(n-h-p)$ sparse features. Hence, the final *Sparse Feature Set (SFS)* is formulated as follows:

$$\text{SFS} = \Big\{T_{sp}^{(1)}, \ldots, T_{sp}^{(N-1)}, t_{sp}^{(1)}, \ldots, t_{sp}^{(n-h-p)}, x_{t-(p-1),d}, \ldots, x_{t-0,d}, x_{t-(p-1),d}^{(\text{nbr, av})}, \ldots, x_{t-0,d}^{(\text{nbr, av})}\Big\}. \tag{7}$$

The final SFS contains $D = (N-1)+(n-h-p)+2p$ features in total.

3.2. *Sparse feature regression*

Since the final sparse feature set has been constructed, we design a simple linear regression model on top of it. In particular, given a set of feature vectors denoted by "fv" $\{F_{i,1}, \ldots, F_{i,D}\}$ of the form represented by Eq. (7), and the corresponding target values $y_i = x_{i+p+h-1,d}$, where $i = 1, 2, \ldots, N_{\text{fv}}$, we define the following linear relationship between them:

$$y_i = \beta_1 F_{i,1} + \cdots + \beta_D F_{i,D}, \tag{8}$$

where $\beta_1, \beta_2, \ldots, \beta_D$ are the D parameters of the model. The N_{fv} feature vectors (and the corresponding target values) are constructed by traversing the set of N available traffic time series of size n each, and selecting the features based on the process described above. After this procedure, $N_{\text{fv}} = N(n-h-p+1) = NZ$ (where $Z = n-h-p+1$) feature vectors have been constructed. The application of Eq. (8) on all feature vectors results in a linear system:

$$F\beta = y, \tag{9}$$

where F is the large sparse $(N_{\text{fv}} \times D)$ coefficient matrix, y is the $(N_{fv} \times 1)$ given vector and β is the $(D \times 1)$ vector of model parameters, and is overdetermined when:

$$N_{\text{fv}} > D \iff NZ > [(N-1)+(n-h-p)+2p], \tag{10}$$

resulting in the following relation:

$$h < n - 3p + 1. \tag{11}$$

In that case, the problem of estimating the parameters of the model is equivalent to solving the optimization problem:

$$\min_{\beta} \|y - F\beta\|_2^2. \tag{12}$$

The above minimization problem has a unique solution, provided that the columns of F are linearly independent, and hence the matrix of normal equations $F^\top F$ is nonsingular. Given that we avoid the dummy variable trap of the sparsification scheme by removing the constant term of the model and that the traffic time series have distinct nonzero values, the regressors of our model are linearly independent. Hence, F is a large sparse full rank matrix. The sparsity pattern of F is

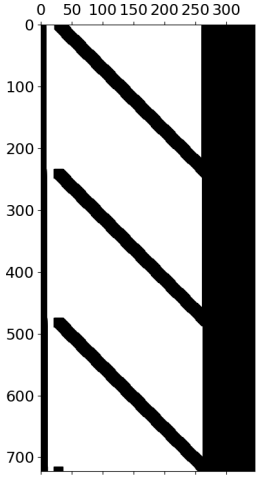

Fig. 1. The sparsity pattern of a block of the coefficient matrix F corresponding to three days of traffic data. The numbers in vertical and horizontal directions represent row and column indices, respectively.

shown in Fig. 1 (more precisely, the figure represents the sparsity pattern of a block of F corresponding to three days of traffic data). The minimization problem described by Eq. (12) is a large sparse least squares problem, which we solve using the *Explicit Preconditioned Conjugate Gradient Least Squares method* (*EPCGLS*) based on sparse approximate pseudoinverse, introduced by Lipitakis *et al.*[26] The overall modelling approach is entitled *Sparse Feature Regression* (*SFR*) and is schematically depicted and algorithmically described in Fig. 2 and Algorithm 1, respectively.

Apart from the base SFR model (referred to hereafter as SFR), we have also designed, implemented and evaluated a variant of our model (denoted by "var"), namely the SFR_{var}. In the SFR_{var} variant, the SFS is enhanced with two additional temporal features, namely T_{d} and T_{w}. The feature T_{d} is the index of the day within a week to which a feature vector belongs. We consider that the week starts on Monday and hence $T_{\text{d}} \in \{1, 2, \ldots, 7\}$. The feature T_{w} represents whether a feature vector comes from a weekday or a weekend day. Hence:

$$T_{\text{w}} = \begin{cases} 1, & \text{if } 1 \leqslant T_{\text{d}} \leqslant 5, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The dense feature set of the SFR_{var} variant is the same as the DFS of SFR, since no new dense features are introduced. The temporal feature T_{w} is already sparse, while the feature T_{d} is sparsified using again the dummy encoding scheme with six categories. For example, the value $T_{\text{d}} = 2$ (i.e., Tuesday) becomes "010000". In this way, the final sparse feature set of SFS_{var} is the following:

$$\text{SFS}_{\text{var}} = \text{SFS} \cup \left\{ T_{\text{d,sp}}^{(1)}, \ldots, T_{\text{d,sp}}^{(6)}, T_{\text{w}} \right\}, \tag{14}$$

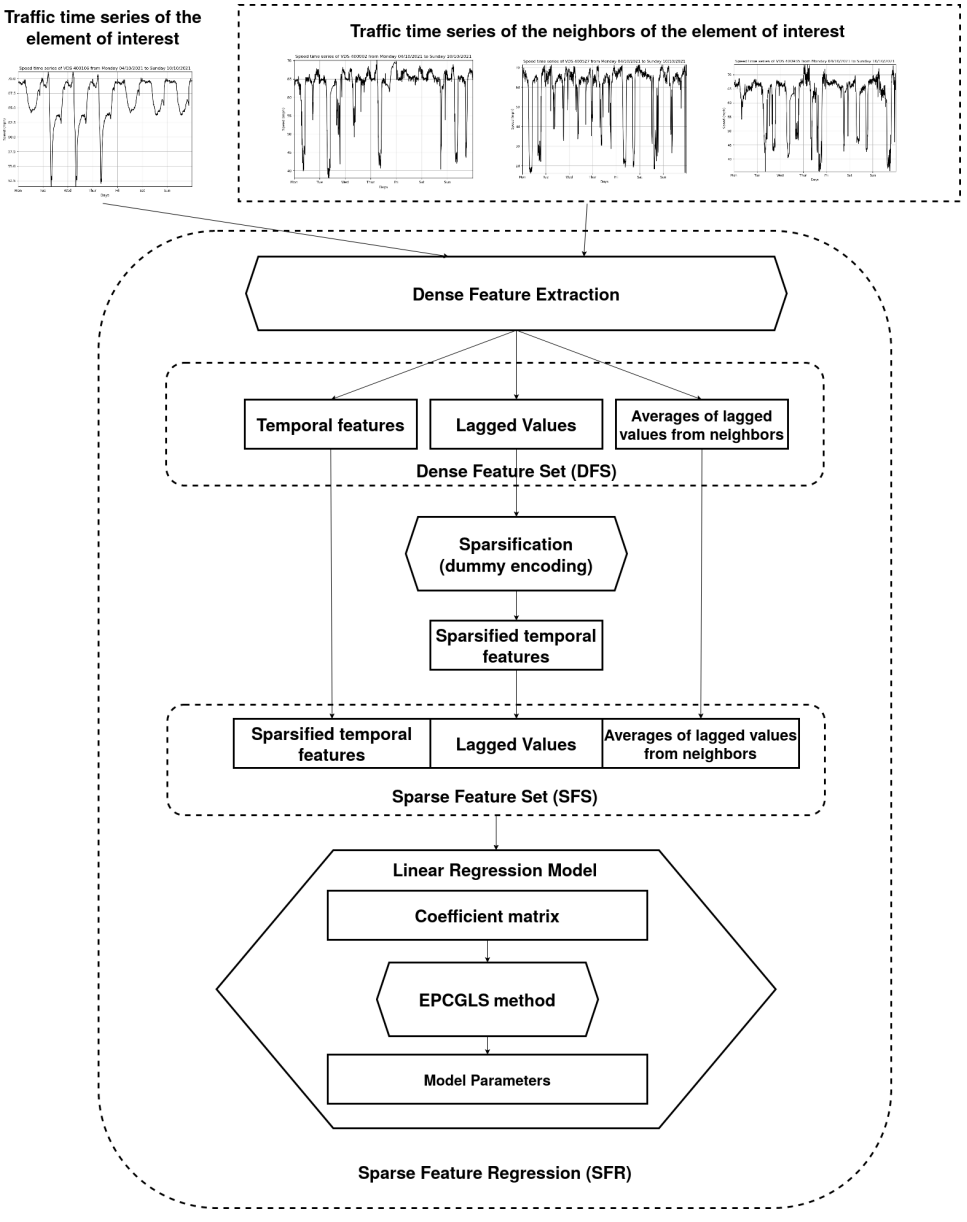

Fig. 2. The SFR modelling approach.

and contains $D = (N-1) + (n-h-p) + 2p + 6 + 1$ features in total. The rest of the modelling approach for SFR_{var} is the same as for SFR. The number of feature vectors and the number of features in DFS and SFS for both SFR and SFR_{var} models are summarized in Table 1.

Algorithm 1 The SFR method

Input: $x_{t,d}$: the values of a traffic time series of a day d with size n
p: the length of a sliding window
h: the forecasting horizon
Output: $\hat{y_i}$: prediction vector

1: **DFS:** $\left\{T, t, x_{t-(p-1),d}, \ldots, x_{t-0,d}, x^{(\text{nbr, av})}_{t-(p-1),d}, \ldots, x^{(\text{nbr, av})}_{t-0,d}\right\}$
2: **Sparsification:** $\left\{T^{(1)}_{sp}, \ldots, T^{(N-1)}_{sp}, t^{(1)}_{sp}, \ldots, t^{(n-h-p)}_{sp}\right\} = DummyEncoding(T, t)$
3: **SFS:** $\left\{T^{(1)}_{sp}, \ldots, T^{(N-1)}_{sp}, t^{(1)}_{sp}, \ldots, t^{(n-h-p)}_{sp}, x_{t-(p-1),d}, \ldots, x_{t-0,d}, x^{(\text{nbr, av})}_{t-(p-1),d}, \ldots, x^{(\text{nbr, av})}_{t-0,d}\right\}$
4: **Linear Regression Model:** $y_i = \beta_1 F_{i,1} + \cdots + \beta_D F_{i,D}$
5: **Resulting Linear System:** $F\beta = y$
6: **Least Squares Problem:** $\min_\beta \|y - F\beta\|_2^2$
7: **EPCGLS Method Based on Pseudoinverse:** $\beta = EPCGLS(F, y)$
8: **Prediction:** $\hat{y_i} = fv_{in}\beta$, where fv_{in} an input feature vector

Table 1. Number of feature vectors and number of features (both in DFS and SFS) for SFR and SFR_{var} models. During the training phase of our model, N_{tr} traffic time series are available (i.e., instead of N).

Model	# Feature Vectors	# Features (DFS)	# Features (SFS)
SFR	$N_{\text{tr}}\, Z$	$1 + 1 + 2p$	$(N_{\text{tr}} - 1) + (Z - 1) + 2p$
SFR_{var}	$N_{\text{tr}}\, Z$	$1 + 1 + 2p + 1 + 1$	$(N_{\text{tr}} - 1) + (Z - 1) + 2p + 6 + 1$

4. Implementation

In this section, we provide the details of the experiments conducted for evaluating the proposed method and report the corresponding results.

4.1. *Datasets*

We used two large-scale real-world traffic datasets from the *Performance Measurement System* (*PeMS*) online data repository.[47] PeMS aggregates data coming from a network of sensors covering all major highways in California, USA. The sensors form clusters named *Vehicle Detection Stations* (*VDS*), which are considered as traffic elements of interest in this work. The first dataset contains speed measurements (in miles per hour – mph) from VDS installed in district 4, i.e., San Francisco Bay Area, California, USA, from a time period of one month, namely October 2021. From this point onwards, we will refer to this dataset as PeMS_s. The second dataset contains flow measurements (in vehicles per 5-minute interval – veh./5m) from VDS installed in district 7, i.e., Los Angeles, California, USA, from a time period of two months, namely October and November 2021. We refer to this dataset hereafter with the term PeMS_f. In both datasets, the data resolution is 5 minutes, which

Table 2. Main characteristics of the PeMS_s and PeMS_f dataset.

Dataset	# VDS (M)	# Time Series Per VDS (N)	Time Series Size (n)	# of Measurements
PeMS_s	100	31	288	892 800
PeMS_f	100	61	288	1756 800

means that each traffic value corresponds to a 5 minute interval. In both datasets, the traffic values are organized into traffic time series of size $n = 288$ each, where each time series corresponds to a day of the total covered period. Hence, for each VDS a set of 31 and 61 time series is constructed for the PeMS_s and PeMS_f dataset, respectively. Each dataset contained missing values when downloaded from PeMS, and therefore we needed to apply a filtering process on the data. In the first filtering round, we kept the VDS for which there were 31 and 61 time series available (for the PeMS_s and PeMS_f dataset, respectively), even incomplete. In the second round, we kept a subset of VDS from the set that emerged from the first round, which contained the VDS that had at least one time series with more than 50% of non-missing values. The overall process resulted in two sets of $M = 100$ VDS for each dataset. The characteristics of both datasets are summarized in Table 2. Example traffic time series from both the PeMS_s and the PeMS_f datasets are shown in Figs. 3 and 4, respectively.

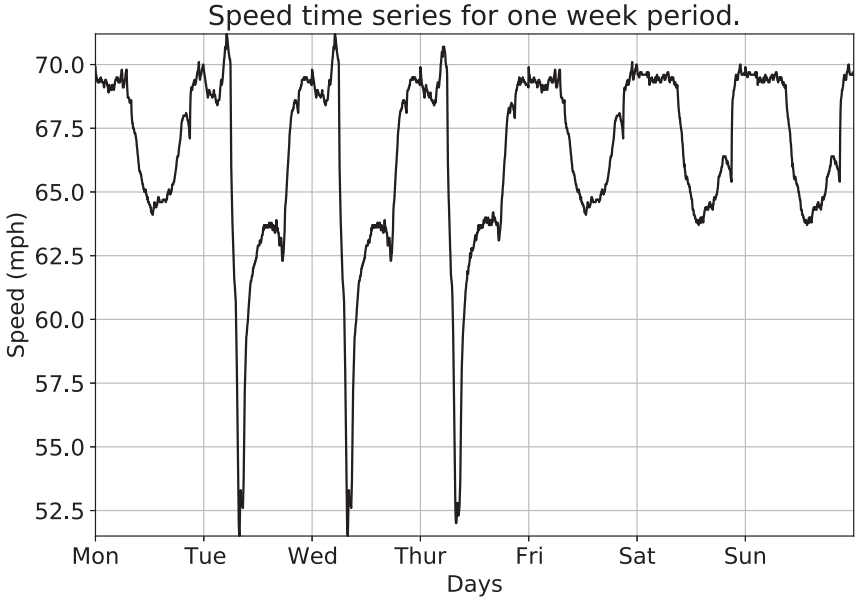

Fig. 3. Example traffic time series from the PeMS_s dataset. The x-axis represents time and the y-axis speed in mph.

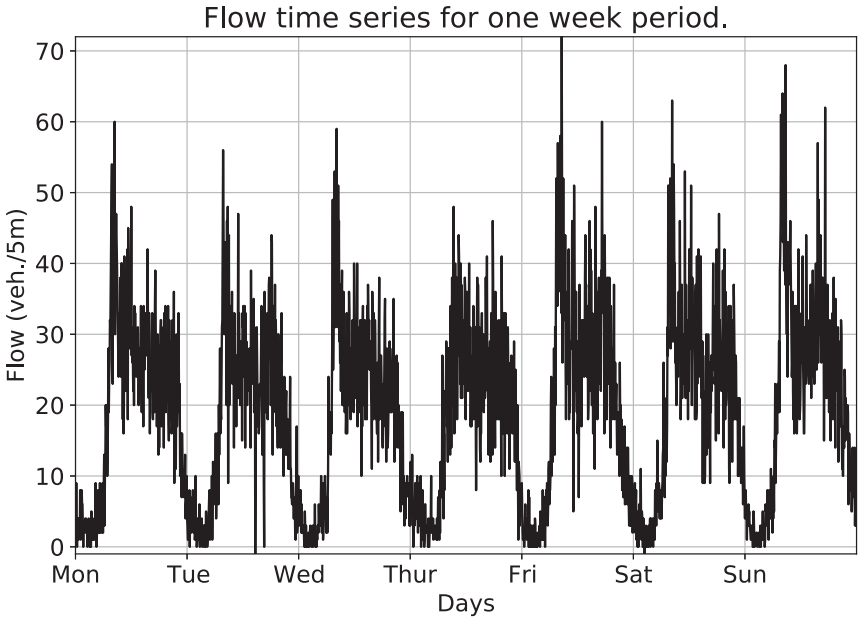

Fig. 4. Example traffic time series from the PeMS$_f$ dataset. The x-axis represents time and the y-axis flow in veh./5min.

4.2. *Setup*

In the paper at hand, the objective is to predict traffic for all elements of interest in an examined network and for several steps ahead in time. In particular, we care about predicting traffic for all elements of our traffic network and not only for specific traffic elements, and we attempt to predict future traffic values for more than 1 step ahead in time. In particular, we considered four forecasting horizons, namely $h \in \{12, 24, 48, 96\}$ steps ahead that correspond to 1, 2, 4 and 8 hours ahead. Given the multi-step forecasting scenario, we apply the direct forecasting strategy,[48,49] hence we build a different instance of a forecasting model for each combination of VDS and forecasting horizon. We have chosen the direct instead of the recursive strategy for multi-step forecasting, since it has been reported in the relevant literature that the latter introduces accumulated error to the predictions.[50,51] The length p of the sliding window rolling over a time series to extract lagged values was set to 36 (i.e., previous 3 hours). In addition, the nonstationary traffic time series from both datasets were transformed to stationary via first-order differencing. Moreover, the k-fold time series cross-validation process was applied.

In particular, the entire time series for each VDS was divided into k folds and for each fold i the first $N_{\text{tr},i}$ traffic time series were used for training and the next N_{val} time series for testing. It holds that $N_{\text{tr},i+1} = N_{\text{tr},i} + N_{\text{val}}, i = 1, \ldots, k-1$. We compute the average value of an error metric across all folds (this is the cross-validation error metric), and then we average these values across all VDS. The

Table 3. k-folds for the PeMS$_s$ dataset.

Fold	Train (%)	Test (%)	N_{tr}	N_{val}
1	First 67.5%	Next 6.5%	21	2
2	First 74%	Next 6.5%	23	2
3	First 80.5%	Next 6.5%	25	2
4	First 87%	Next 6.5%	27	2
5	First 93.5%	Next 6.5%	29	2

Table 4. k-folds for the PeMS$_f$ dataset.

Fold	Train (%)	Test (%)	N_{tr}	N_{val}
1	First 75%	Next 5%	46	3
2	First 80%	Next 5%	49	3
3	First 85%	Next 5%	52	3
4	First 90%	Next 5%	55	3
5	First 95%	Next 5%	58	3

resulting final value is the average cross-validation error metric that is used to compare the proposed method with benchmarks. The parameters of the k-fold time series cross-validation process for the PeMS$_s$ and PeMS$_f$ dataset are presented in Tables 3 and 4, respectively. The number of neighboring elements N_{nbr} for the SFR models was set to 3. The experiments were performed on a personal computer having an Intel Core i7-7700HQ CPU with 8 cores at 2.80 GHz and 8 GB RAM. In addition, the proposed method has been implemented in C++ programming language and used the Intel MKL library of mathematical routines. Furthermore, the benchmark models were implemented using the scikit-learn, LightGBM, TensorFlow and Facebook Prophet Python libraries. Moreover, the total execution time was reduced by utilizing the proposed approach by Salamanis *et al.*[52]

In order to quantify the forecasting accuracy of the traffic forecasting models, we used the *Root-Mean-Square Error* (*RMSE*) and the *Mean Absolute Percentage Error* (*MAPE*) metrics:

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{pr}}} \sum_{i=1}^{N_{\text{pr}}} (y_i - \hat{y}_i)^2}, \tag{15}$$

$$\text{MAPE} = \frac{100\%}{N_{\text{pr}}} \sum_{i=1}^{N_{\text{pr}}} \left| \frac{y_i - \hat{y}_i}{y_i} \right|, \tag{16}$$

where y_i is the real traffic value, $\hat{y}_i$ is the predicted traffic value and N_{pr} is the number of predictions. RMSE and MAPE values were computed for each fold of the k-fold time series cross-validation process and then these values have been averaged for all folds resulting in the *cross-validation RMSE* and *cross-validation MAPE* values, respectively. Then, the average of the cross-validation RMSE and

cross-validation MAPE values across all VDS were computed resulting in the final *average cross-validation RMSE* and *average cross-validation MAPE* values. These were the values used for comparing the several methods with each other in terms of forecasting accuracy. Moreover, the computational efficiency of the forecasting methods was evaluated by measuring the total time (in seconds) for executing the k-fold time series cross validation process for all VDS.

Furthermore, we combined the forecasting accuracy results with the performance results using the *Adjusted Ratio of Ratios* (*ARR*) metric. ARR was first introduced by Brazdil *et al.*,[53] who used it to evaluate the accuracy of different classification algorithms on a set of datasets in relation to the corresponding execution times. Considering two traffic forecasting models fm_1 and fm_2 evaluated on a traffic dataset td, the ARR metric for them is defined as follows:

$$\mathrm{ARR}^{\mathrm{td}}_{(\mathrm{fm}_1,\,\mathrm{fm}_2)} = \frac{\mathrm{FA}^{\mathrm{td}}_{\mathrm{fm}_1}/\mathrm{FA}^{\mathrm{td}}_{\mathrm{fm}_2}}{1+\mathrm{AccD}\log\left(\mathrm{P}^{\mathrm{td}}_{\mathrm{fm}_1}/\mathrm{P}^{\mathrm{td}}_{\mathrm{fm}_2}\right)}, \tag{17}$$

where $\mathrm{FA}^{\mathrm{td}}_{\mathrm{fm}_i}$ and $\mathrm{P}^{\mathrm{td}}_{\mathrm{fm}_i}$ is the forecasting accuracy and the execution time of the forecasting model fm_i on the traffic dataset td, respectively. In the paper at hand, the forecasting accuracy is defined as $\mathrm{FA}^{\mathrm{td}}_{\mathrm{fm}_i} = 100 - \mathrm{MAPE}^{\mathrm{td}}_{\mathrm{fm}_i}$, where $\mathrm{MAPE}^{\mathrm{td}}_{\mathrm{fm}_i}$ refers to the average cross-validation MAPE value of the forecasting model fm_i on the dataset td. In addition, "AccD" is a free parameter expressed as a percentage that controls the balance between the accuracy of the predictions and the time required for producing them. For example, AccD = 4% can be explained as trading 4% accuracy for a four-fold speedup. Given N_m traffic forecasting models evaluated on N_{td} traffic datasets, a separate ARR value can be estimated for each forecasting model fm_i as follows:

$$\mathrm{ARR}_{\mathrm{fm}_i} = \frac{\displaystyle\sum_{\mathrm{fm}_j}\left[\prod_{\mathrm{td}_j}\mathrm{ARR}^{\mathrm{td}_j}_{(\mathrm{fm}_i),\,(\mathrm{fm}_j)}\right]^{\frac{1}{N_{\mathrm{td}}}}}{N_m}. \tag{18}$$

A high $\mathrm{ARR}_{\mathrm{fm}_i}$ value indicates a high rank for fm_i model. Hence, we computed the values of ARR for both the proposed and the benchmark forecasting models separately for PeMS_s and PeMS_f datasets, i.e., $N_{\mathrm{td}} = 1$ in Eq. (18).

The proposed method (both base and variant) was compared with five benchmark traffic forecasting models, namely (1) the *Autoregressive Integrated Moving Average* (*ARIMA*) model, (i.e., ARIMA$(p, 1, 0)$ model[54]), (2) the *Prophet model* (proposed by Taylor and Letham[55] and implemented by Facebook), (3) the *Long Short-Term Memory* (*LSTM*) model,[5] (4) the Gradient Boosting Decision Tree (GBDT) model and (5) the GBDT model in combination with the Gradient-based One-Side Sampling (GOSS) training acceleration technique[56] (GBDT-GOSS). ARIMA was used as a benchmark given its extensive application in several traffic forecasting scenarios reported in literature, while the Prophet model was chosen

given its ability to fit time series with nonlinear trends and multiple seasonality. From the deep learning category of traffic forecasting models, the LSTM model was selected. In particular, in the PeMS_s dataset, the LSTM architecture consisted of 1 LSTM layer of 8 units and the Swish activation function, i.e., $f(x) = x\,\text{sigmoid}(\beta x)$[57] and an output layer with 1 unit and linear activation function. It was trained using the RMSProp optimizer and MAPE as objective function with batch size 64 and 10 epochs. The same configuration was also applied to the PeMS_f dataset. Moreover, the gradient-boosting-based models GBDT and GBDT-GOSS with 100 estimators each were used in order to check how well the proposed model performs compared to ensemble models.

5. Results

In this section, the results of the experiments conducted for the evaluation of the proposed method in terms of both forecasting accuracy and computational efficiency are presented. In particular, the results for the PeMS_s dataset are presented in Tables 5–8, while the results for the PeMS_f dataset are presented in Tables 9–12. In Tables 5 and 9, the labels "RMSE" and "MAPE" refer to the average cross-validation RMSE and average cross-validation MAPE, respectively.

In the PeMS_s dataset, the proposed method presents the best balance between accuracy of predictions and time required for achieving these predictions for all forecasting horizons. In particular, even though it produces less accurate predictions compared to the gradient-boosting-based models at a level between 1% and 2%, it achieves this result in approximately half the time compared to the gradient-boosting-based models. This means that when taking into account the exchange

Table 5. Prediction errors of all models in PeMS_s.

Method	$h = 12$		$h = 24$		$h = 48$		$h = 96$	
	MAPE (%)	RMSE (mph)	MAPE (%)	RMSE (mph)	MAPE (%)	RMSE (mph)	MAPE (%)	RMSE (mph)
ARIMA	7.581	5.745	11.509	7.869	16.355	9.917	17.153	9.605
Prophet	10.854	6.753	11.201	6.878	11.769	7.062	11.145	6.491
LSTM	14.074	10.162	19.392	13.237	23.374	14.946	25.7	16.264
GBDT	6.497	4.708	8.542	5.746	10.348	6.434	9.529	5.833
GBDT-GOSS	6.669	4.767	8.737	5.833	10.524	6.551	9.745	5.935
SFR	7.254	4.882	9.567	6.074	11.701	7.146	11.112	6.677
SFR_{var}	7.303	4.894	9.632	6.104	11.74	7.168	10.885	6.515

Table 6. Ranking of the evaluated traffic forecasting models on the PeMS_s dataset across all forecasting horizons.

Method	GBDT	GBDT-GOSS	SFR	SFR_{var}	Prophet	ARIMA	LSTM
Rank	1.0	2.0	3.25	3.75	5.25	5.75	7

Table 7. Computational efficiency of all models in PeMS$_s$.

Method	Performance (sec.)			
	$h = 12$	$h = 24$	$h = 48$	$h = 96$
ARIMA	72.584	66.23	65.193	55.585
Prophet	3253.195	3170.624	3165.491	3194.974
LSTM	1972.209	1858.135	1685.268	1369.849
GBDT	62.408	67.517	66.678	56.644
GBDT-GOSS	68.968	64.91	56.531	46.884
SFR	32.479	30.157	25.961	18.436
SFR$_{\text{var}}$	33.777	31.815	27.533	19.926

Table 8. ARR of all models in PeMS$_s$.

Method	$h = 12$		$h = 24$		$h = 48$		$h = 96$	
	AccD (4%)	AccD (10%)	AccD (4%)	AccD (10%)	AccD (4%)	AccD (10%)	AccD (4%)	AccD (10%)
ARIMA	1.032	1.066	1.017	1.052	0.984	1.016	0.974	1.006
Prophet	0.917	0.846	0.945	0.873	0.969	0.894	0.975	0.894
LSTM	0.887	0.829	0.852	0.797	0.83	0.777	0.801	0.751
GBDT	1.049	1.089	1.056	1.092	1.066	1.1	1.078	1.112
GBDT-GOSS	1.045	1.081	1.055	1.092	1.068	1.107	1.08	1.12
SFR	**1.054**	**1.117**	**1.06**	**1.126**	**1.068**	**1.137**	**1.081**	**1.158**
SFR$_{\text{var}}$	**1.052**	**1.114**	**1.058**	**1.121**	**1.067**	**1.133**	**1.083**	**1.156**

Table 9. Prediction errors of all models in PeMS$_f$.

Method	$h = 12$		$h = 24$		$h = 48$		$h = 96$	
	MAPE (%)	RMSE (veh./5m)	MAPE (%)	RMSE (veh./5m)	MAPE (%)	RMSE (veh./5m)	MAPE (%)	RMSE (veh./5m)
ARIMA	29.865	31.234	41.038	47.77	60.488	68.776	78.87	84.421
PROPHET	33.835	34.637	32.1	34.828	27.924	31.434	26.317	28.313
LSTM	75.724	111.506	89.766	146.143	82.131	138.283	87.395	144.053
GBDT	25.03	21.731	28.532	26.458	32.613	30.522	32.062	28.945
GBDT-GOSS	25.181	21.954	28.745	26.789	32.829	30.862	32.306	29.265
SFR	27.725	23.777	31.57	30.707	31.678	34.123	27.654	30.601
SFR$_{\text{var}}$	27.875	24.011	31.849	31.202	32.119	34.796	27.719	30.767

of forecasting accuracy for speedup, the proposed method achieves better balance in both low (i.e., AccD = 4%) and high (i.e., AccD = 10%) tradeoff level. This is an important outcome since the proposed traffic forecasting model is a simple linear regression model that nevertheless manages to achieve high levels of accuracy compared to complex ensemble models. In addition, SFR models achieve better forecasting accuracy results compared to the two statistical benchmark models (i.e., ARIMA and Prophet) in all cases. While ARIMA is a quite efficient model given

Table 10. Ranking of the evaluated traffic forecasting models on the PeMS_f dataset across all forecasting horizons.

Method	SFR	GBDT	Prophet	GBDT-GOSS	SFR_{var}	ARIMA	LSTM
Rank	2.5	2.5	3.25	3.5	3.5	5.75	7

Table 11. Computational efficiency of all models in PeMS_f.

Method	Performance (sec.)			
	$h = 12$	$h = 24$	$h = 48$	$h = 96$
ARIMA	79.81	76.328	75.903	65.676
Prophet	5951.744	5995.088	5821.995	5864.654
LSTM	2219.266	2197.65	1994.552	1543.403
GBDT	87.156	90.673	85.53	71.077
GBDT-GOSS	91.586	90.781	88.633	69.653
SFR	106.277	101.902	93.74	72.161
SFR_{var}	132.451	132.417	116.047	89.542

Table 12. ARR of all models in PeMS_f.

Method	$h = 12$		$h = 24$		$h = 48$		$h = 96$	
	AccD (4%)	AccD (10%)	AccD (4%)	AccD (10%)	AccD (4%)	AccD (10%)	AccD (4%)	AccD (10%)
ARIMA	1.341	1.423	1.743	1.874	0.88	0.935	0.549	0.586
Prophet	1.153	1.078	1.871	1.772	1.605	1.501	2.144	1.997
LSTM	0.326	0.309	0.146	0.138	0.281	0.265	0.238	0.224
GBDT	**1.442**	**1.526**	**2.14**	**2.286**	1.615	1.702	2.132	2.243
GBDT-GOSS	**1.438**	**1.518**	**2.133**	**2.278**	1.608	1.693	2.124	2.237
SFR	1.379	1.45	2.038	2.168	**1.637**	**1.72**	**2.28**	**2.398**
SFR_{var}	1.37	1.43	2.018	2.13	**1.618**	1.689	**2.269**	**2.369**

its simplicity, its forecasting performance deteriorates as the forecasting horizon increases. This does not happen with SFR, whose forecasting performance remains quite stable as the forecasting horizon increases, mainly due to its increased capacity in terms of parameters. On the other hand, the Prophet model presents a behavior quite similar to the SFR models in terms of forecasting accuracy, yet it is quite inefficient given that it attempts to model all changes in the trend of the time series as well as its multiple seasonalities. Moreover, LSTM produces acceptable but not so accurate predictions, given enough time for hyperparameter tuning through trial and error. In addition, the LSTM models take a quite long time to train. These results are consistent with the general understanding that deep learning models can achieve very good prediction results, given enough available resources, namely data, computational resources and time for tuning and training.

The ranking of the models in terms of forecasting accuracy, presented in Table 6, resulted from the application of the *Friedman nonparametric statistical test.*[58] In particular, we evaluated the null hypothesis H_0 in order that the means of the average cross-validation MAPE values across all forecasting horizons of two or more models are the same. With a significance level $a = 5\%$, we rejected H_0 in all cases. However, by applying the post-hoc *Bonferroni-Dunn*[59] test we failed to reject the null hypothesis H_0 that there is statistically significant difference between the GBDT models and the SFR models at a significance level of 5%. This means that even though the GBDT models achieve lower forecasting errors compared to the SFR models, the difference between the two types of models is marginal.

In the PeMS_f dataset, the proposed method presents the best tradeoff between prediction error and execution time for the long-term forecasting horizons (i.e., $h = 48$ and $h = 96$), while it achieves the second best balance after the GBDT models for the short-term horizons (i.e., $h = 12$ and $h = 24$). In particular, the gradient-boosting-based models produce more accurate predictions compared to the SFR models in the short-term forecasting horizons with a difference in MAPE around 2.5%, while on the contrary they produce less accurate predictions compared to the SFR models in the long-term horizons with a difference in MAPE again around 2.5%. In terms of performance, the gradient-boosting-based models are approximately 10 seconds faster compared to the SFR models in the short-term forecasting horizons, while this difference shrinks as h increases. Hence, by considering accuracy and performance, the GBDT models outperform the SFR models in both high and low tradeoff levels for the short-term forecasting horizons, while the situation is reversed in the long-term horizon case. Therefore, it becomes evident that the proposed method is more suitable for long-term forecasting in time series with high volatility as it is the case for the flow time series of the PeMS_f dataset.

Regarding the other benchmarks, the situation is similar with that of the PeMS_s dataset. In addition, we performed also the nonparametric Friedman test for evaluating the null hypothesis H_0 that the predictions obtained are not statistically different. With a significance level $a = 5\%$, we rejected the H_0. The ranking of the models in terms of forecasting accuracy produced by the Friedman test is presented in Table 10. Moreover, by performing the Bonferroni-Dunn post-hoc test we failed to reject the null hypothesis H_0 that there is statistically significant difference between the GBDT models and the SFR models at a significance level of 5%. Similarly to the case of the PeMS_s dataset, this result means that the difference in forecasting accuracy between the gradient-boosting-based models and the proposed method is marginal.

5.1. *Discussion*

As shown by all the above results, the SFR models achieve good forecasting accuracy for both datasets and all forecasting horizons, and this happens mainly due to the appropriate selection and processing of the features. In particular, the temporal

features represent the specific moments in the past on which the model should focus in order to predict the value of a traffic variable at a specific moment in the future. The actual values of the traffic variable at these specific moments in the past are given by the lagged values. Additionally, the spatiotemporal correlations of the element of interest with other elements in the network are considered through the averages of the neighbors' lagged values. Moreover, the sparsification of the temporal features increases the parameterization of the SFR models, which in turn leads to better prediction accuracy. However, this direction of investigating why increased parameterization improves the prediction accuracy of models on unseen data, is an open research problem that in recent years has attracted the interest of researchers.[60,61]

Therefore, in future work we shall investigate in more detail the properties of the SFR models (mainly the properties of the covariance matrix), in order to better understand the effect of increased parameterization on forecasting accuracy. Moreover, increasing the capacity in terms of parameters of the SFR models not in any way but specifically through the sparsification of appropriate features leads to efficient training by leveraging the computational advantages of iterative methods used for solving large sparse least squares problems.

Furthermore, the proposed method can be considered as *explainable* in the sense that one can understand which features affect more the produced predictions by examining the top D_{top} features of the SFR models through bar diagrams like those presented in Figs. 5 and 6 for the PeMS_s and PeMS_f dataset, respectively. In particular, in the x-axis the indices of the top features are provided, while in

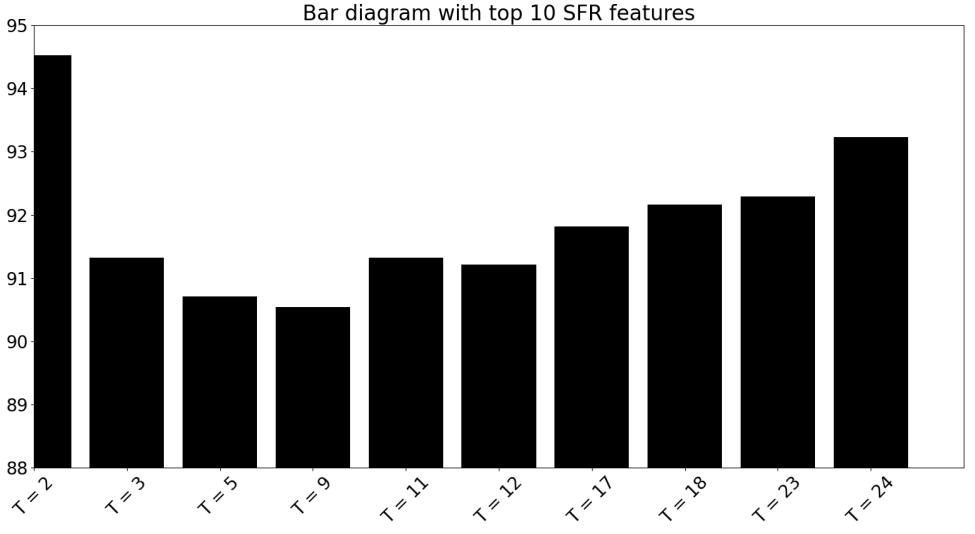

Fig. 5. Bar diagram of the top 10 SFR features for a VDS from the PeMS_s dataset (VDS 400000, $h = 12$). The x-axis represents the indices of the top features and the y-axis their corresponding feature importance scores.

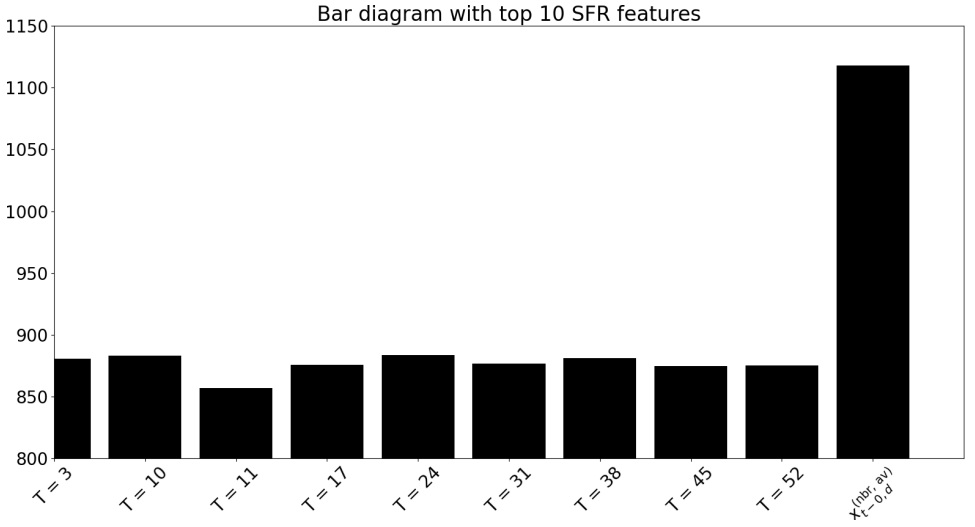

Fig. 6. Bar diagram of the top 10 SFR features for a VDS from the PeMS$_f$ dataset (VDS 715898, $h = 12$). The x-axis represents the indices of the top features and the y-axis their corresponding feature importance scores.

the y-axis their corresponding feature importance scores measured by the absolute value of the corresponding parameters of the SFR models are provided. In order to compute those feature importance scores, the dense features of the models have been scaled into the range $[0, 1]$ using the min-max scaling method prior to fitting the SFR models. Through the visual inspection of these bar diagrams it is understood that (for the examined VDS) in the case of the PeMS$_s$ dataset the temporal feature T (i.e., Eq. (2)) is the most important one, while in the case of the PeMS$_f$ dataset both the temporal feature T and the lagged value $x_{t-0,d}^{(\mathrm{nbr,\,av})}$ (i.e., Eq. (4)–(5)) are the most importance features for the SFR models.

6. Synopsis and Concluding Remarks

In the paper at hand, we presented a new traffic prediction method. Spatiotemporal features are carefully extracted from the available traffic time series, some of them are sparsified, and a linear regression model is designed using the final sparse feature set. The parameters of the model are computed by solving an optimization problem. We used two real-world datasets and performed large-scale experiments in order to evaluate the accuracy and speed of the proposed method. The results of the experiments indicated that the proposed method exhibits good balance between prediction accuracy and speed in all cases. This is an important result because it verifies our intuition that cleverly selected features, overparameterization through sparsification and simple linear modelling are key elements for both accurate and fast large-scale traffic forecasting.

Future work includes the analysis of the properties of the proposed model in order to better understand the effect of increased parameterization on forecasting accuracy. Moreover, the effect of additional new temporal features on the prediction accuracy and speed of the proposed method will be investigated. Furthermore, the proposed method will be evaluated on more datasets and forecasting scenarios.

Acknowledgments

The authors declare that they have no conflict of interest. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. The authors would like to thank the anonymous reviewers for their constructive comments.

References

1. Y. Zou, G. Shi, H. Shi and H. Zhao, Traffic incident classification at intersections based on image sequences by HMM/SVM classifiers, *Multimedia Tools Appl.* **52** (2011) 133–145.
2. G. Fusco, C. Colombaroni and N. Isaenko, Short-term speed predictions exploiting big data on large urban road networks, *Transportation Research Part C*: *Emerging Technologies* **73** (2016) 183–201.
3. B. Yu, H. Yin and Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, in *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence* (*IJCAI'18*) (AAAI Press, 2018), pp. 3634–3640.
4. T. Mallick, P. Balaprakash, E. Rask and J. Macfarlane, Graph-partitioning-based diffusion convolutional recurrent neural network for large-scale traffic forecasting, *Transportation Research Record* **2674**(9) (2020) 473–488.
5. Y. Li, R. Yu, C. Shahabi and Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, in *Int. Conf. on Learning Representations* (*ICLR*) (2018), pp. 1–14.
6. Y. Huang, Y. Weng, S. Yu and X. Chen, Diffusion convolutional recurrent neural network with rank influence learning for traffic forecasting, in *2019 18th IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications/13th IEEE Int. Conf. on Big Data Science and Engineering* (*TrustCom/BigDataSE*) (IEEE Computer Society, Los Alamitos, CA, USA, August 2019), pp. 678–685.
7. A. Stathopoulos and M. G. Karlaftis, A multivariate state space approach for urban traffic flow modeling and prediction, *Transportation Research Part C*: *Emerging Technologies* **11**(2) (2003) 121–135.
8. B. Williams and L. Hoel, Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results, *Journal of Transportation Engineering* **129** (11 2003) 664–672.
9. D. D. Billings and J.-S. Yang, Application of the arima models to urban roadway travel time prediction — A case study, in *2006 IEEE Int. Conf. on Systems, Man and Cybernetics*, Vol. 3 (2006), pp. 2529–2534.
10. X. Min, J. Hu and Z. Zhang, Urban traffic network modeling and short-term traffic flow forecasting based on gstarima model, in *13th Int. IEEE Conf. on Intelligent Transportation Systems* (2010), pp. 1535–1540.

11. Y.-J. Wu, F. Chen, C.-T. Lu and S. Yang, Urban traffic flow prediction using a spatio-temporal random effects model, *Journal of Intelligent Transportation Systems* **20**(3) (2016) 282–293.
12. J. Guo and B. M. Williams, Real-time short-term traffic speed level forecasting and uncertainty quantification using layered Kalman filters, *Transportation Research Record* **2175**(1) (2010) 28–37.
13. H. Liu, J. W. C. van Lint, H. J. van Zuylen and K. Zhang, Two distinct ways of using Kalman filters to predict urban arterial travel time, in *2006 IEEE Intelligent Transportation Systems Conference* (2006), pp. 845–850.
14. B. Zhou, D. He and Z. Sun, Traffic modeling and prediction using ARIMA/GARCH model, in *Modeling and Simulation Tools for Emerging Telecommunication Networks*, eds. A. Nejat Ince and E. Topuz (Springer US, Boston, MA, 2006), pp. 101–121.
15. N. Anand, C. Scoglio and B. Natarajan, GARCH — Non-linear time series model for traffic modeling and prediction, in *2008 IEEE Network Operations and Management Symposium* (*NOMS 2008*) (2008), pp. 694–697.
16. C. Chen, J. Hu, Q. Meng and Y. Zhang, Short-time traffic flow prediction with ARIMA/GARCH model, in *2011 IEEE Intelligent Vehicles Symposium* (*IV*) (2011), pp. 607–612.
17. F. Schimbinschi, L. Moreira-Matias, V. X. Nguyen and J. Bailey, Topology-regularized universal vector autoregression for traffic forecasting in large urban areas, *Expert Systems with Applications* **82** (2017) 301–316.
18. T. Zhou, D. Jiang, Z. Lin, G. Han, X. Xu and J. Qin, Hybrid dual Kalman filtering model for short-term traffic flow forecasting, *IET Intelligent Transport Systems* (2019).
19. D. Pavlyuk, Short-term traffic forecasting using multivariate autoregressive models, *Procedia Engineering* **178** (2017) 57–66 (Part of Special Issue: *RelStat-2016*: *Proc. of the 16th Int. Scientific Conference Reliability and Statistics in Transportation and Communication*) (October 19–22, 2016) (Transport and Telecommunication Institute, Riga, Latvia).
20. S. Cheng, F. Lu, P. Peng and S. Wu, Short-term traffic forecasting: An adaptive ST-KNN model that considers spatial heterogeneity, *Computers, Environment and Urban Systems* **71** (2018) 186–198.
21. B. Sun, W. Cheng, P. Goswami and G. Bai, Short-term traffic forecasting using self-adjusting k-nearest neighbours, *IET Intelligent Transport Systems* **12** (2018) 41–48.
22. Z. Liu, W. Du, D.-m. Yan, G. Chai and J. Guo, Short-term traffic flow forecasting based on combination of k-nearest neighbor and support vector regression, *Journal of Highway and Transportation Research and Development* (English edn.) **12** (2018) 89–96.
23. L. Cai, Y. Yu, S. Zhang, Y. Song, Z. Xiong and T. Zhou, A sample-rebalanced outlier-rejected k-nearest neighbor regression model for short-term traffic flow forecasting, *IEEE Access* **8** (2020) 22686–22696.
24. H. Hong, W. Huang, X. Xing, X. Zhou, H. Lu, K. Bian and K. Xie, Hybrid multi-metric k-nearest neighbor regression for traffic flow prediction, in *2015 IEEE 18th Int. Conf. on Intelligent Transportation Systems* (2015), pp. 2262–2267.
25. S. Kolidakis, G. Botzoris, V. Profillidis and P. Lemonakis, Road traffic forecasting — A hybrid approach combining artificial neural network with singular spectrum analysis, *Economic Analysis and Policy* **64** (2019) 159–171.
26. A.-D. E. Lipitakis, C. K. Filelis-Papadopoulos, G. A. Gravvanis and D. Anagnostopoulos, A class of generic approximate sparse pseudoinverse matrix technique based

on incomplete QR factorization, in *7th Annual Conf. on Computational Science & Computational Intelligence* (*CSCI'20*) (2020), pp. 1341–1347.
27. Q. Luo and Y. Zhou, Spatial-temporal structures of deep learning models for traffic flow forecasting: A survey, in *2021 4th Int. Conf. on Intelligent Autonomous Systems* (*ICoIAS*) (2021), pp. 187–193.
28. KH. N. Bui, J. Cho and H. Yi, Spatial-temporal graph neural network for traffic forecasting: An overview and open research issues, *Applied Intelligence* (2021).
29. Z. Lu, W. Lv, Y. Cao, Z. Xie, H. Peng and B. Du, LSTM variants meet graph neural networks for road speed prediction, *Neurocomputing* **400** (2020) 34–45.
30. Z. Lu, W. Lv, Z. Xie, B. Du and R. Huang, Leveraging graph neural network with LSTM for traffic speed prediction, in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation* (*SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI*) (2019), pp. 74–81.
31. X. Wang, Y. Ma, Y. Wang, W. Jin, X. Wang, J. Tang, C. Jia and J. Yu, Traffic flow prediction via spatial temporal graph neural network, in *Proc. of The Web Conference 2020* (*WWW '20*) (Association for Computing Machinery, New York, NY, USA, 2020), pp. 1082–1092.
32. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, Attention is all you need, in *Proc. of the 31st Int. Conf. on Neural Information Processing Systems* (*NIPS'17*) (Curran Associates Inc., Red Hook, NY, USA, 2017), pp. 6000–6010.
33. T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, in *Int. Conf. on Learning Representations* (*ICLR*) (2017), pp. 1–14.
34. Z. Diao, X. Wang, D. Zhang, Y. Liu, K. Xie and S. He, Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting, in *Proc. of the AAAI Conf. on Artificial Intelligence*, Vol. 33 (2019), pp. 890–897.
35. Q. Zhang, J. Chang, G. Meng, S. Xiang and C. Pan, Spatio-temporal graph structure learning for traffic forecasting, in *Proc. of the AAAI Conf. on Artificial Intelligence*, Vol. 34 (2020), pp. 1177–1185.
36. W. Chen, L. Chen, Y. Xie, W. Cao, Y. Gao and X. Feng, Multi-range attentive bicomponent graph convolutional network for traffic forecasting, in *The Thirty-Fourth AAAI Conf. on Artificial Intelligence* (*AAAI 2020*), *The Thirty-Second Innovative Applications of Artificial Intelligence Conference* (*IAAI 2020*), *The Tenth AAAI Symp. on Educational Advances in Artificial Intelligence* (*EAAI 2020*) (New York, NY, USA, February 7–12, 2020) (AAAI Press, 2020), pp. 3529–3536.
37. Y. Zhang, T. Cheng, Y. Ren and K. Xie, A novel residual graph convolution deep learning model for short-term network-based traffic forecasting, *International Journal of Geographical Information Science* **34**(5) (2020) 969–995.
38. Z. Cui, K. Henrickson, R. Ke and Y. Wang, Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting, *IEEE Transactions on Intelligent Transportation Systems* **21**(11) (2020) 4883–4894.
39. L. Chen, K. Han, Q. Yin and Z. Cao, GDCRN: Global diffusion convolutional residual network for traffic flow prediction, in *Knowledge Science, Engineering and Management*, eds. G. Li, H. T. Shen, Y. Yuan, X. Wang, H. Liu and X. Zhao (Springer International Publishing, Cham, 2020), pp. 438–449.

40. S.-S. Kim, M. Chung and Y.-K. Kim, Urban traffic prediction using congestion diffusion model, in *2020 IEEE Int. Conf. on Consumer Electronics - Asia* (*ICCE-Asia*) (2020), pp. 1–4.
41. C. Park, C. Lee, H. Bahng, Y. Tae, S. Jin, K. Kim, S. Ko and J. Choo, ST-GRAT: A novel spatio-temporal graph attention networks for accurately forecasting dynamically changing road speed, in *Proc. of the 29th ACM Int. Conf. on Information and Knowledge Management* (Association for Computing Machinery, New York, NY, USA, 2020), pp. 1215–1224.
42. X. Kong, W. Xing, X. Wei, P. Bao, J. Zhang and W. Lu, STGAT: Spatial-temporal graph attention networks for traffic flow forecasting, *IEEE Access* **8** (2020) 134363–134372.
43. K. Tian, J. Guo, K. Ye and C.-Z. Xu, ST-MGAT: Spatial-temporal multi-head graph attention networks for traffic forecasting, in *2020 IEEE 32nd Int. Conf. on Tools with Artificial Intelligence* (*ICTAI*) (2020), pp. 714–721.
44. X. Kong, J. Zhang, X. Wei, W. Xing and W. Lu, Adaptive spatial-temporal graph attention networks for traffic flow forecasting, *Applied Intelligence* (2021).
45. G. Guo and W. Yuan, Short-term traffic speed forecasting based on graph attention temporal convolutional networks, *Neurocomputing* **410** (2020) 387–393.
46. D. B. Suits, Use of dummy variables in regression equations, *Journal of the American Statistical Association* **52**(280) (1957) 548–551.
47. PeMS, Caltrans freeway performance measurement system (2022), Last accessed: 20-01-2022.
48. A. Sorjamaa, J. Hao, N. Reyhani, Y. Ji and A. Lendasse, Methodology for long-term prediction of time series, *Neurocomputing* **70**(16) (2007) 2861–2869 (Neural Network Applications in Electrical Engineering. Selected papers from the 3rd Int. Work-Conf. on Artificial Neural Networks (*IWANN 2005*)).
49. C. Hamzacebi, D. Akay and F. Kutay, Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting, *Expert Systems with Applications* **36**(2, Part 2) (2009) 3839–3844.
50. S. Masum, Y. Liu and J. Chiverton, Comparative analysis of the outcomes of differing time series forecasting strategies, in *2017 13th Int. Conf. on Natural Computation, Fuzzy Systems and Knowledge Discovery* (*ICNC-FSKD*) (2017), pp. 1964–1968.
51. L. M. B. Maggi, *Multiscale Forecasting Models* (2018).
52. A. Salamanis, G. Gravvanis, C. Filelis-Papadopoulos and D. Tzovaras, An efficient method for accelerating training of short-term traffic prediction models in large-scale traffic networks, in *24th Pan-Hellenic Conf. on Informatics* (*PCI 2020*) (Association for Computing Machinery, New York, NY, USA, 2020), pp. 289–294.
53. P. Brazdil, C. Soares and J. Costa, Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results, *Machine Learning* **50** (2003) 251–277.
54. T. Alghamdi, K. Elgazzar, M. Bayoumi, T. Sharaf and S. Shah, Forecasting traffic congestion using ARIMA modeling, in *2019 15th Int. Wireless Communications & Mobile Computing Conference* (*IWCMC*) (2019), pp. 1227–1232.
55. S. J. Taylor and B. Letham, Forecasting at scale, *The American Statistician* **72** (2018) 37–45.
56. G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu, LightGBM: A highly efficient gradient boosting decision tree, in *Advances in Neural Information Processing Systems*, eds. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, Vol. 30 (Curran Associates, Inc., 2017), pp. 1–9.

57. P. Ramachandran, B. Zoph and Q. V. Le, Searching for activation functions, *CoRR* **abs/1710.05941** (2017).
58. M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, *Journal of the American Statistical Association* **32**(200) (1937) 675–701.
59. O. J. Dunn, Multiple comparisons among means, *Journal of the American Statistical Association* **56**(293) (1961) 52–64.
60. M. Belkin, S. Ma and S. Mandal, To understand deep learning we need to understand kernel learning, in *Proc. of the 35th Int. Conf. on Machine Learning* (PMLR, 10–15 July 2018), eds. J. Dy and A. Krause, Vol. 80, pp. 541–549.
61. P. L. Bartlett, P. M. Long, G. Lugosi and A. Tsigler, Benign overfitting in linear regression, *Proc. of the National Academy of Sciences* **117**(48) (2020) 30063–30070.