Transportation Research Part C 112 (2020) 62–77

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

A graph CNN-LSTM neural network for short and long-term traffic forecasting based on trajectory data☆

Toon Bogaerts[a,b,*], Antonio D. Masegosa[a,c], Juan S. Angarita-Zapata[a], Enrique Onieva[a], Peter Hellinckx[b]

[a] *DeustoTech, Faculty of Engineering, University of Deusto, Av. Universidades, 24, 48007 Bilbao, Spain*
[b] *University of Antwerp - imec, IDLab - Faculty of Applied Engineering, Sint-Pietersvliet 7, 2000 Antwerp, Belgium*
[c] *IKERBASQUE, Basque Foundation for Science, 48011 Bilbao, Spain*

ARTICLE INFO

Keywords:
Deep learning
Graph convolutional network
LSTM
Traffic forecasting
Trajectory data
GPS data
Long term
Short term
ITS

ABSTRACT

Traffic forecasting is an important research area in Intelligent Transportation Systems that is focused on anticipating traffic in order to mitigate congestion. In this work we propose a deep neural network that simultaneously extracts the spatial features of traffic, using graph convolution, and its temporal features by means of Long Short Term Memory (LSTM) cells to make both short-term and long-term predictions. The model is trained and tested using sparse trajectory (GPS) data coming from the ride-hailing service of DiDi in the cities of Xi'an and Chengdu in China. Besides, presenting the deep neural network, we also propose a data-reduction technique based on temporal correlation to select the most relevant road links to be used as input. Combining the suggested approaches, our model obtains better results compared to high-performance algorithms for traffic forecasting, such as LSTM or the algorithms presented in the TRANSFOR19 forecasting competition. The model is capable of maintaining its performance over different time-horizons from 5 min to up to 4 h with multi-step predictions.

1. Introduction

Traffic congestion generates economic, social and environmental issues in many cities around the world. Therefore, the main objective of traffic managers is to handle traffic in a way that enables a higher quality of service on the roads to guarantee better mobility and less congestion. In the past, traffic managers were limited to formulate and deploy reactive traffic response plans to dealt with traffic congestion once it was present on the roads without any effort for anticipating it. Nevertheless, the use of Information and Communication Technologies and the Internet of Things has helped to build Intelligent Transportation Systems (ITSs), which in turn has allowed the use of Traffic Forecasting (TF) methods to predict traffic conditions on the roads.

Over the last decades, TF has been a hot research topic in the area of ITSs due to its strategical advantage to anticipate and prevent traffic congestion. The main objective of TF is the prediction of traffic measures (e.g., travel time, traffic flow) in the near future, ranging from the next few minutes to up to several hours, based on historical traffic data (Vlahogianni et al., 2014).

The increasing variety and volume of available traffic data provided by ITSs have caused a shift in the way TF is approached, switching from a traffic theory-based perspective to a data-driven one. Within data-driven approaches, statistical and Machine

☆ This article belongs to the Virtual Special Issue on "TRANSFOR 19".

* Corresponding author at: University of Antwerp - imec, IDLab - Faculty of Applied Engineering, Sint-Pietersvliet 7, 2000 Antwerp, Belgium

E-mail addresses: toon.bogaerts@uantwerpen.be (T. Bogaerts), ad.masegosa@deusto.es (A.D. Masegosa), js.angarita@deusto.es (J.S. Angarita-Zapata), enrique.onieva@deusto.es (E. Onieva), peter.hellinckx@uantwerpen.be (P. Hellinckx).

https://doi.org/10.1016/j.trc.2020.01.010
Received 30 June 2019; Received in revised form 26 December 2019; Accepted 11 January 2020
Available online 29 January 2020

Learning (ML) methods are the two main categories. In the early stages of data-driven methods applied to TF, we can find a wide variety of literature on statistical approaches, such as ARIMA models (Hamed et al., 1995); however, they have shown some shortcomings when dealing with complex TF problems. For this reason, most of the current literature in this area is focused on ML methods because of their better ability to address high-dimensional data and extracting non-linear relationships.

Some well-known examples of traditional ML methods applied to TF are Support Vector Machines (SVM), k-Nearest Neighbors (k-NN) (Cai et al., 2016; Habtemichael and Cetin, 2016) or Random Forest, among others (Lana et al., 2018; Vlahogianni et al., 2014). Despite their success, classical ML methods present limitations when capturing the spatial and temporal relationships of traffic patterns (Yang, 2013; Zhao et al., 2018; Zhang et al., 2016). In recent years, Deep Neural Networks (DNNs) have caught the attention of transportation research because of their high representational power that facilitates dealing with the aforementioned components of traffic and their good performance in terms of accuracy and error metrics (Angarita-Zapata et al., 2019; Do et al., 2018; Ermagun and Levinson, 2018).

Some early examples of the application of DNNs to TF are Deep Belief Networks (Huang et al., 2014) and stacked auto-encoders (Lv et al., 2015); nevertheless, research soon shifted to the use other types of DNNs, such as Recurrent Neural Networks (RNNs) (Cui et al., 2018b; Yu et al., 2017b) or Convolutional Neural Networks (CNNs) (Ma et al., 2015; Zhang et al., 2017). The main reason behind the use of RNNs is their ability to capture short and long temporal dependencies of traffic data, especially LSTM and Gated Recurrent Unit (GRU) approaches. In the case of CNNs, the main motivation behind their application is their capacity to model spatial relationships of traffic. Notwithstanding, given that conventional CNNs were not originally designed to deal with graph-structured data, as is the case with traffic information and roadnetworks, researchers developed variants to better capture spatial relationships that are commonly known as Graph CNNs (Defferrard et al., 2016; Atwood and Towsley, 2016).

Over recent years, we have observed the appearance of hybrid architectures that combine RNNs and Graph CNNs with the aim of capturing simultaneously the spatial and temporal relationships of traffic; these approaches have demonstrated a more superior performance than their ”pure” RNN and Graph CNN counterparts (Cui et al., 2018b; Yu et al., 2017b; Jin et al., 2018; Zhao et al., 2018). However, as far as we know, previous studies on hybrid Graph CNN-RNN architectures were designed to work with point-wise traffic sensors or a reduced set of road links for limiting the complexity and dimension of the input data. Furthermore, they are only tested for short-term time horizons, which correspond to less than 60-minutes-ahead predictions according to the categorisations proposed in recent literature (Lana et al., 2018; Angarita-Zapata et al., 2019). Regarding the former issue, when considering the whole road network it is important to exploit the strengths of DNNs in order to deal with the particularities of trajectory data, since, in this case, traffic information is spread around the whole city and the temporal dependencies between road links are not explicitly represented on the data. As for the latter issue, dealing with short-term horizons is the common approach of ML and DNNs, leaving the long-term predictions as an open issue in the transportation literature (Lana et al., 2018).

Keeping the aforementioned challenges in mind, the main contributions of this paper are as follow:

- An efficient data reduction technique for selecting the most relevant road links for both short- and long-term TF.
- A novel Graph CNN-RNN architecture for TF based on trajectory data with two variants that differ mainly in the CNN's layer disposal. One that has the usual alternation of convolution and max-pooling layers, and another one in which max pooling is applied before convolution in order to further reduce the dimension of the input data without losing relevant information.

The proposed GraphCNN-LSTM model is validated using data from DiDi Chuxing Gaia Open Data Initiative, which supported the Transportation Forecasting Competition (TRANSFOR19) organized by the Standing Committee on Artificial Intelligence, the Advanced Computing Applications (ABJ70) of the Transportation Research Board, and the IEEE ITS Technical Activities Sub-Committee “Smart Cities and Smart Mobility”. An initial version of this model was presented in the TRANSFOR19 competition, ranking third.[1]

The rest of the paper is structured as follows. Section 2 presents related work in the area of DNN methods applied to TF. Section 3 details the proposed data reduction method and the DNN architecture. Then, Sections 5 and 6 summarize the experimental set-up and result analysis, respectively. Finally, Section 7 presents the main conclusions of this work.

2. Related work

The aim of this section is to review recent literature related to different methodologies for TF, with a special focus on DNN approaches, and to highlights the main novelties of our proposal w.r.t. these approaches.

TF can be tackled using different modelling perspectives. The four most common approaches found in transportation literature are: (i) the statistical time-series perspective (Ermagun and Levinson, 2018), (ii) the supervised regression problem (Angarita-Zapata et al., 2019; Howell, 2018), (iii) the supervised classification problem (Angarita-Zapata et al., 2018; Lopez-Garcia et al., 2016), and (iv) a clustering-pattern recognition approach (Aldhyani and Joshi, 2018). They are described as follows.

First, the statistical time-series perspective is based on developing models that fit observations made at prior times and using them to predict future traffic (Pavlyuk, 2017; Karimpour et al., 2017; Li et al., 2015). This is at the expense of being able to explain why a specific prediction was made and understanding the underlying causes behind the predictions (Brownlee, 2018). Secondly, the supervised regression approach is focused on building a predictive model without prior models or error distribution specifications

[1] https://sites.google.com/site/trbcommitteeabj70/abj70s-latest-news/another-successful-trb-for-abj70.

and using historical data to predict a continuous traffic variable (e.g., traffic speed) based on unseen data (Ermagun and Levinson, 2018).

Contrary to the supervised regression problem, the supervised classification approach focuses on forecasting discrete traffic values (e.g., level of service) based on continuous traffic historical data (Angarita-Zapata et al., 2018; Ma et al., 2015; Shi and Abdel-Aty, 2015). It is important to clarify that the forecasting of discrete variables could be also addressed as a regression problem predicting a continuous traffic measure and then categorizing these predictions to obtain discrete outputs.

Lastly, the fourth modelling approach is the clustering-pattern recognition problem. It is focused on investigating the dynamic traffic relationships of different locations. This is achieved by characterizing similar traffic measure values from one road to another and then grouping the locations in clusters that divide the road network into correlated groups (Triguero et al., 2017). This is possible due to traffic's correlations in the temporal and spatial domain. Exploiting such similarities enables traffic conditions to be predicted cluster by cluster for future times, based on historical traffic data.

This paper is focused on the supervised regression approach. The literature on TF reports a wide variety of ML methods for supervised traffic prediction such as Neural Networks, SVM or Random Forest (Angarita-Zapata et al., 2019; Lana et al., 2018). These ML methods have shown satisfactory results when the complexity of the problem is moderate. However, they present important drawbacks when simultaneously capturing the spatial and temporal correlations of traffic data; particularly when the dimensionality of the problem is high, as it is the case with TF based on trajectory data. In this sense, DNN approaches have demonstrated superiority in finding these spatio-temporal patterns w.r.t. traditional ML methods.

One of the first contributions of DNN to forecast traffic dates back to 2015 when Lv et al. (2015) proposed a Deep Belief Network that detected both spatial and temporal correlations in data used to predict traffic. Three years later, Yu et al. (2017b) implemented a conventional CNN architecture to convert traffic speed data into a series of static images from which traffic predictions were made. In spite of the novelty of considering CNNs for TF, the image representation of traffic data sometimes could not capture the real physical characteristics of the network under study, which in turn led to making predictions that did not correspond to the actual traffic conditions on the road.

Later in 2018, Cui et al. (2018b) proposed a DNN architecture based on LSTM that captures the spatial features of traffic data in small freeway and urban environments. The authors thus opened a path to exploring the potential of LSTMs to approach TF in bigger network areas, which can include the full range of spatial dependencies among all road segments. Furthermore, they discussed possible enhancements for the proposed model by incorporating non-traffic information into the input data. In concordance with this last issue, Polson and Sokolov (2017) developed a DNN approach that modelled the nonlinear spatio-temporal effects in recurrent and non-recurrent traffic congestion patterns. These patterns include data about construction zones, weather, special events and traffic incidents. Nevertheless, the model presented by the authors was focused on traffic data coming from point-wise sensors, limiting the spatial coverage of the predictions.

Improvements were made by (Cui et al., 2018b,a; Polson and Sokolov, 2017; Yu et al., 2017a) who developed Graph CNN-LSTM methods to learn the spatial and temporal relationship between roads in an entire network in order to make predictions. Although these previous DNN methods can learn such spatial dependencies, they tend to be over-complex and inevitably capture a certain amount of noise and spurious relationships. Then, Duan et al. (2018) presented a hybrid NN to predict urban flow that combines a CNN to extract the spatial features of traffic data together with a LSTM that obtains the temporal dimension of traffic. Unlike the work done by Cui et al. (2018a) and Yu et al. (2017a), Duan et al. (2018) introduced a model to predict traffic flow in each cell of a grid that represents different areas of the city; however, as was the case with the study presented by Yu et al. (2017b), this approach is not suitable to capture the actual spatial relationships of the road network.

More recently, Liu et al. (2018) added non-traffic data to the input layer of a DNN architecture. Its main novelty relies on its ability to reduce the accuracy degradation caused by missing data. In the same year, Zhao et al. (2018) proposed a novel DNN that mixes a temporal Graph CNN component with a GRU. This architecture is used to capture the topological structure of the road network in a more realistic way and to model the spatial dependence among the traffic in different locations.

The main differences of the approach presented in this paper and the ones listed above rely on three main points: (1) the data reduction method used to select the most relevant road links in the network for the TF problem at hand, given that previous papers used either road links from a predefined region or just an ad-hoc selection; (2) a new Graph CNN-LSTM architecture for TF with a different CNN layer disposal that considerably reduces the number of network parameters; and (3) the consideration of longer time horizons for the predictions, more concretely of up to four hours (previous approaches do not exceed time horizons of sixty minutes).

3. Data pre-processing workflow and proposed data reduction method

In order to facilitate the understanding of the role and functioning of the proposed data reduction method, and in turn, the proposed network architecture, this section describes the data pre-processing workflow (Section 3.1) in which the data reduction method is one of its last stages. The objective of this workflow is to convert raw GPS data trajectories to traffic data in a matrix data format, which is the format in which CNNs ingest the data. Given that we have proposed a Graph CNN, in the first stage the output of the workflow is an adjacency matrix. Within this context, the data reduction method is in charge of choosing the most representative links in the road network in order to decrease the size of the aforementioned matrix and to make it more manageable for the Graph CNN. As one of the main contributions of this work, the proposed data reduction method is described in more detail in Section 3.2.1).

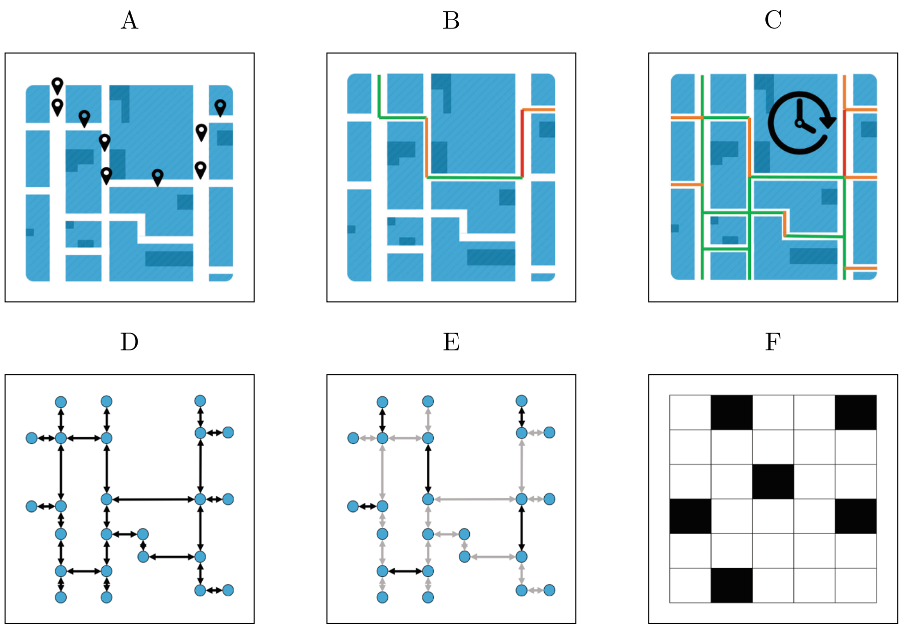

Fig. 1. Scheme of the data pre-processing workflow: (A) Original GPS track points of a single trip; (B) Map-matched track of GPS points, (C) Aggregation of all map-matched trips, this is done with five-minute intervals; (D) Creation of a graph structure based on the road-network; (E) selection of most significant links in this structure; (F) Translation from the most significant links to an array-like structure like the adjacency matrix.

3.1. Data pre-processing workflow

This section presents the pre-processing pipeline designed to obtain an adjacency matrix with traffic information from the raw GPS data of vehicle trajectories. The stages of the designed workflow are common when Graph-CNNs are applied over GPS data to predict traffic, and are as follows (Fig. 1): (i) Map-matching, which matches the GPS-traces to their corresponding road links; (ii) Data aggregation, which combines individual entries to average entries; (iii) Data Imputation which deals with missing values; (iv) Data reduction which selects a subset with the most relevant road links in order to decrease the size of the resulting adjacency matrix; and (v) the construction of the adjacency matrices. In the following sections, each of these steps are explained in more detail.

3.1.1. Map-matching

The map-matching process consists of finding the most probable sequence of road links followed by a vehicle, given the GPS trace of that vehicle. The process is shown in steps A and B of Fig. 1. This sequence is usually noisy and can be inaccurate. In our approach, we used the map-matching method proposed by Newson and Krumm based on Hidden Markov Models (Newson and Krumm, 2009). In this method, the observations correspond to the GPS measurements and the probability that a GPS measurement corresponds to a specific link is inverse to the distance to that road link, which is calculated using a Gaussian function. Moreover, the Viterbi algorithm (Forney, 1973) was used to find the most probable sequence in the Hidden Markov Model that, in turn, corresponds to the most probable sequence of road links for that GPS trace. In this way, a sequence of temporally annotated GPS points is converted to a sequence of temporally annotated road links with the associated average speed in that link.

3.1.2. Data aggregation

Given that carrying out traffic forecasting with the different approaches used in this paper requires traffic information at a road link level, a process is needed to change the sequences of temporally annotated road links (resulting from the map-matching stage), to traffic information per road link and time interval. This process is illustrated in Fig. 1C. Since the individual entities of map-matched data contain information on the average speed of a specific vehicle in the corresponding road segment, the aggregation of these entities per road link and time interval give rise to the desired information. In our case, the length of the time interval was set to five minutes (so we have a total of $12 \times 24 = 288$ intervals per day), and the aggregation process provides the average speed and the number of different vehicles within the time interval for each road link.

3.1.3. Data imputation

Missing data is a common issue when dealing with GPS-data coming from ride-hailing services. As there are fewer requests at night, the available data during this timespan is sparse, meaning that some intervals within this period have no data for some road links. This issue has more impact on non-primary roads as they usually have a small number of samples.

In our paper, the missing data is filled using two different approaches. On the one hand, for small gaps of a maximum of two consecutive steps (time intervals), linear interpolation is used among the following and previous steps. On the other hand, if the gap of information is larger than two steps, an average-based imputation method is used. This average is calculated on the available training data. At first, the average is calculated on the values within the same five-minute interval and day of the week. In case these values are missing as well, the value is set to the average in the same time interval, considering the whole week; otherwise, the average of the corresponding 15 min is used; and otherwise, the average over the same hour and weekday is considered. After this pre-processing state, the data has a fixed size, with one entry per road segment and time interval.

3.1.4. Data reduction

As explained above, the objective of the data reduction stage is to select the subset of the most relevant road links in order to reduce the size of the adjacency matrix resulting from the pre-processing pipeline and thus making its processing by Graph-CNN networks more manageable (in terms of computation) and speeding up its training process. An abstraction of this process is shown in Fig. 1D and E. Performing data reduction always implies a trade-off between the aforementioned advantages and the loss of information. When exploring urban traffic data, it is possible that numerous roads might contain similar information. Think of a traffic jam at an intersection: the congestion can be identified on all connecting roads.

The specific approach used in this study will be explained in Section 3.2.1, as one of the contributions of our paper.

3.2. Construction of adjacency matrix

In this last stage, an adjacency matrix is extracted for the subset of links selected in the previous step. Fig. 1E and F illustrate this process. Given a graph $G = (V, E)$, where V is the vertex set (intersections) and E is the edges (road links), the matrix extracted from the selected road segments $U \subset V$, is a square matrix with dimensions $|Vt(U)| \times |Vt(U)|$, where $Vt(N)$ represents the vertices connected to the segments included in U. For each traffic measure considered (feature), a different adjacency matrix is extracted. These matrices have identical shapes and are stacked to form a three-order tensor with features on the third axis.

Given that CNN assumes that cells in the adjacency matrix that are closed in the input tensor are related, the rows and columns of the adjacency matrix are sorted according to the latitude and longitude of the corresponding vertices, respectively. In this way, cells that are closed in the adjacency matrix will represent road links that are close in the real road network. This approach results in a sparse matrix with some clusters representing key road clusters in the city as shown in Fig. 2. In this way, the Graph CNNs can extract more meaningful spatial patterns.

3.2.1. Proposed data reduction method

The proposed data reduction technique for selecting the subset U of the most relevant road links, works based on the correlation between the objective segment and the subset U. The main idea behind this is simple: the higher the absolute correlation of the U links with the target road segment, the more predictive capacity they will contribute to the model to make predictions at the selected location. Keeping this idea in mind, we have designed two variants of the data reduction method:

- *Simple Correlation*: In this case, the Pearson correlation coefficient is calculated between the target link $o \in E$ and each one of the other links $E' = \{E - o\}$ at the same time instant. Concretely, the following formula is used to compute the correlation for each segment is used:

$$R_e = \sum_{t \in T} \sum_{i \in M} Corr(m_i(o, t), m_i(e, t)) \tag{1}$$

where $e \in E'$, T is the set of time intervals considered, M is the set of traffic measurements considered, $Corr$ is the Pearson's correlation coefficient, and $m_i(x, t)$ is the value of the $i - th$ measurement for road link x at time interval t. Then, the n segments with the

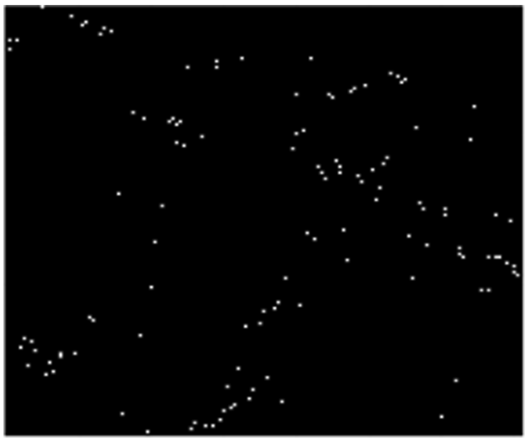

Fig. 2. Example of a sorted adjacency matrix in the city of Xian.

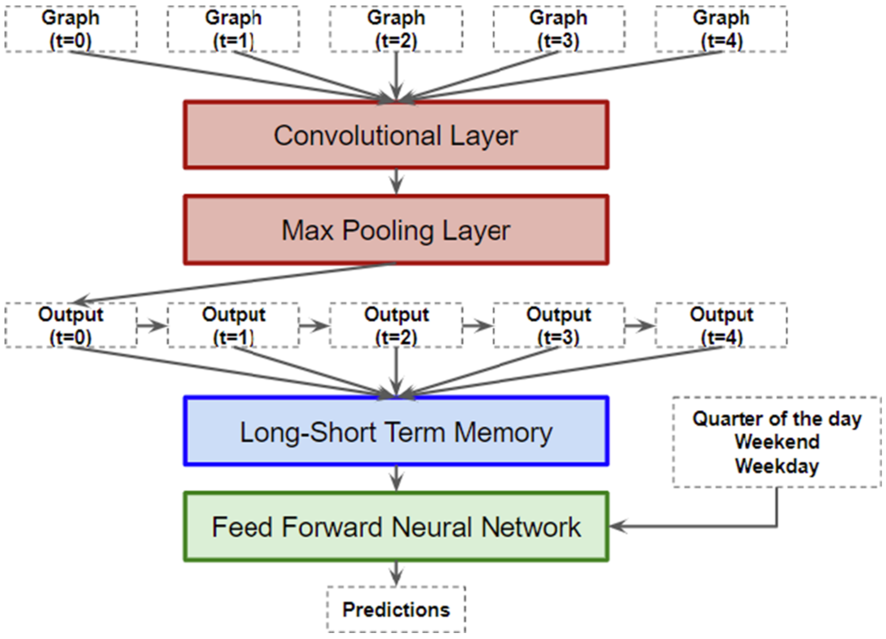

Fig. 3. Network architecture.

highest R values are selected as being the most relevant, that is, as the set of vertices that will define the set U.

- *Time-based correlation*: In this variant, the correlation is also calculated between the target link $o \in E$ and each of the other links E', but instead of considering just the current time interval, the correlation is also computed over each pair (prediction horizon, previous time step). The following equation describes this process:

$$TR_{eh} = \sum_{t \in T} \sum_{s \in S} \sum_{i \in M} Corr(m_i(o, t+h), m_i(e, t-s)), \tag{2}$$

where $h \in H$ is the time horizon considered for the prediction, and S is the set of past time intervals considered for computing the TR value. In this case, in order to select the n most relevant segments, the $\left\lfloor \frac{n}{|H|} \right\rfloor$ road links with the highest TR_{eh} are selected for each time horizon considered in the prediction task, where $\lfloor x \rfloor$ indicates the integer part. If the n segments are not completed as such, the remaining $n - |H| \cdot \left\lfloor \frac{n}{|H|} \right\rfloor$ edges with the highest R_e values are selected.

4. Proposed network architecture

The second contribution of this paper is a hybrid DNN architecture for TF that aims at finding both spatial and temporal relationships in traffic data. Concretely, it is divided in three main components that are shown in Fig. 3: (i) a Graph CNN, (ii) a RNN based on LSTM cells, and (iii) a Feed-Forward Neural Network (FFNN). Generally, the hybrid DNN uses historical information by considering sequences of the m previous time steps of data provided using a graph's adjacency matrix for each time step. These structures are generated in order to able to consider spatial dependencies within traffic data. Once the spatial information is processed, the recurrent unit is fed with the resulting tensor (a second order tensor of $m \times p$ dimension, where p is established by the characteristics of the network); this stage evaluates the previous time steps of the spatial relationships with the aim of extracting possible temporal trends. Finally, results are gathered into a FFNN and concatenated with optional contextual data (e.g., quarter of the day, weekend, weekday) to produce the final output. The blocks that compose the main architecture are presented and described in the following subsections.

4.1. Graph convolutional neural network

Extracting spatial dependencies from traffic data is a challenge in TF. Traditional CNNs are able to detect spatial relations in data types such as images or other two or three-dimensional data structures. However, the complexity of a city's road network can make it difficult or even impossible to represent this information in a two-dimensional matrix. These complex road networks can be represented with the use of graph structures, where streets are edges and intersections are nodes. Fig. 4 shows a graph structure of Xi'an city with only primary and secondary roads. There are two general approaches to transform these graph structures into a structured data format. On the one hand, to modify the spectral domain by using Fourier transforms (Bruna et al., 2013) and on the other hand, to expand the spatial definition of the graph structure (Niepert et al., 2016). In this work, we use the latter. This transformation

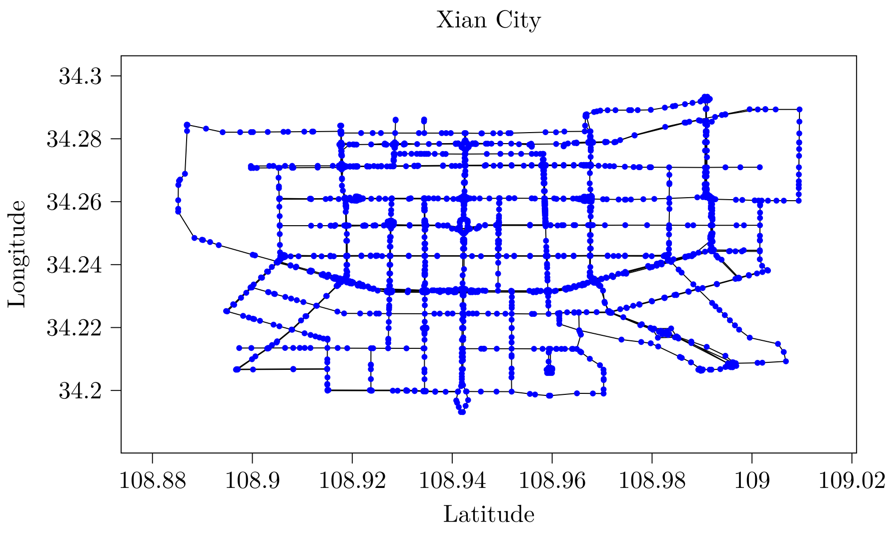

Fig. 4. Graph structure of the primary and secondary roads in the city of Xi’an.

consists of constructing an adjacency matrix from the graph structure. This matrix can be interpreted by a CNN and extra features can be created with stacked adjacency matrices to extend the feature space. Using this approach, spatial features of complex road networks can be codified in a way that can be processed by CNNs.

In this study, two different CNN architectures are used:

- *CMCM) a double sequence of convolution layers followed by* max-*pooling layer*. In this strategy, low-level features are extracted first, followed by the first pair of convolution/max-pooling layers, whereas medium-level features are computed from these low-level features, followed by the second pair of convolution/max-pooling layers.
- *(MMCCM) two* max-*pooling layers followed by two convolutional layers and one* max-*pooling layer*. Given that the adjacency matrices have a sparse distribution of data due to the characteristics of road networks (as we saw in Fig. 2), this second approach first uses max-pooling to reduce the size of the adjacency matrix by 'zooming in'. This process is shown in Fig. 5. As can be seen, spatial

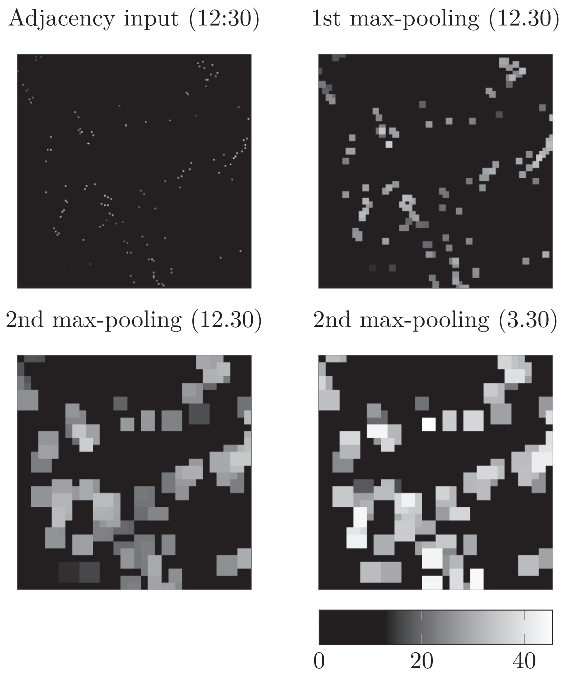

Fig. 5. Example of an adjacency matrix displaying the average speed [km/h] on 1 October 2016. The first row illustrates the raw adjacency matrix and the result of the first application of the max-pooling layer at 12.30. The second row shows the application of the second max-pooling layer at 12.30 and at 3.30.

patterns are more visible and, for example, we can clearly see that the overall average speed at night (2nd max-pooling 3.30) is higher, resulting in a lighter adjacency matrix. The main advantage of this approach is the decrease in the number of trainable parameters of the network. This reduction does not drastically decrease the number of trainable parameters in the Graph CNN, but it is significant in the connection between the convolution- and recurrent sub-models, dividing approximately the total number of network parameters by a factor of 10. This reduction results in faster training, reduced complexity and better generalization capabilities for the network.

Fig. 5 also helps understand how convolution layers extract temporal patterns. During the first convolution, individual clusters are grouped to form more unified clusters. During the second stage, combinations of segments that are most relevant to the predicted segment are extracted by the convolution. The last layer uses max pooling to reduce the size of the tensor. In this example, the input to the pooling layer contains mainly the same information as the output, which is a great conversion as the last tensor occupies less memory.

4.2. *Long-short term memory neural network*

Similar to spatial dependencies, temporal relationships are key features to consider in TF. They can be extracted from sequential data with the use of RNNs; however, these networks are highly complex and have difficulty finding long-term dependencies in the data (Pascanu et al., 2013). Because of these drawbacks, a LSTM network is used. This type of networks are more robust to exploding-vanishing gradients and have proven capable of extracting long-term dependencies from sequential data, a very important ability for TF.

4.3. *Feed forward neural network*

In this part of the proposed architecture, each neuron in a layer is connected to all the neurons in the layer. Stacking these layers reveals higher dimensional relationships within the input data. Regarding forecasting, these networks can be useful when evaluating and combining the extracted features with some contextual information. This extra information can contain a variety of features such as time-related features, which give the model a sense of time, or historical data, which helps the network to detect some patterns, for instance rush hours or weekends.

4.4. *Temporal graph convolutional network*

With the aim of taking advantage of the combination of both spatial and temporal relationships, as was shown in Fig. 3, a sequential model is constructed to: (1) extract spatial patterns by feeding stacked adjacency matrices of different time intervals into the Graph CNN; (2) extract the time-related trends from the aforementioned spatial features by stacking them according to time and then feeding them into the LSTM network (making it possible to consider both temporal and spatial relationships at the same time); and (3) make predictions by combining spatio-temporal patterns with extra features containing contextual information and inputting them into the FFNN.

5. Experimental framework

This section describes the main aspects of the experimentation carried out in this study. Concretely, it presents the two data-sets used with traffic information from the cities of Xi’an and Chengdu in China; the hyper-parameter configuration for the considered variants of our network architecture; and the experimental set-up including the state-of-the-art methods used for comparison purposes together with the performance measures previously considered.

5.1. *Data-set description*

GPS-data taken from the GAIA initiative of DiDi company has been used to test the proposed architecture. The complete data set used in this study can be requested from the official GAIA initiative website.[2] This data set contains trajectories of DiDi Express and DiDi Premier drivers within the cities of Xi'an and Chengdu. The data contains trips from October to November 2016. The sample period of the measured GPS track points is approximately two to four seconds. All the information regarding the request and trip is anonymized. This data-set was used as the data set for the TRANSFOR 2019 traffic forecasting competition. As is the case with most of the available GPS data-sets, it contains only a fraction of the total number of vehicles in the city, but it will be considered as ground truth data in this study. The total size of the data-set in Comma-Separated Values (CSV) format is 36.5 GB. From these two months of data, the first seven weeks are used as training data and the last week as the test set.

In order to consider different TF scenarios, multiple road segments are used as target data from both Xi'an and Chengdu. Table 1 shows the properties of the five selected segments from each city with two extra segments in Xi’an. These two extra segments (2748, 160) were target road links in the TRANSFOR19 competition. These segments are chosen based on their standard deviation and speed

[2] https://outreach.didichuxing.com/research/opendata/en/.

Table 1
Segment selected for traffic prediction in Xi'an and Cheng Du. rush_mean and rush_std refers to the mean speed and standard deviation for that segment in rush hours (6.00–11.00 and 16.00–21.00). (*) Denotes segments used in the TRANSFOR19 forecasting competition.

Chengdu	Mean [km/h]	std	rush_mean [km/h]	rush_std	samples	Missing [%]
207,941	25,28	3,53	24,43	3,02	187,677	5
7171	28,38	4,66	27,42	4,31	130,234	9
9273	54,25	11,69	49,77	13,96	187,936	6
210,521	23,39	9,11	20,64	9,72	147,525	9
128	26,91	5,95	25,97	5,68	130,881	12
Xian						
161	26,16	8,87	22,62	6,50	139,258	17
2747	23,67	4,93	22,89	4,33	190,621	16
92,053	22,07	3,73	20,79	3,21	203,041	16
7532	32,23	11,97	26,97	10,32	237,328	16
7524	36,74	8,70	33,43	9,74	159,451	20
160*	16,92	6,50	15,15	5,44	137,235	17
2748*	25,43	9,80	21,46	7,76	170,771	17

distribution to cover different traffic profiles. Other criteria the segments should fulfill are whether they are primary or secondary roads, have a length of over 100 m, have less than 20% missing values and more than 130 k measurements to guarantee the quality of the data used to train the models.

5.2. Network hyper-parameters

Tables 2 and 3 show the hyper-parameters of the Graph CNN variants and the whole network, respectively. They have been established according to standard parameters in the literature and preliminary experiments. Regarding Graph CNN variants, the configuration displayed for the MMCCM variant permits going from having roughly 19 million parameters using the first approach, to 1.5 million using the second approach.

As for the whole network hyper-parameters, the tensors' size can slightly differ depending on the shape of the input adjacency matrices. Five variants of the main architecture are evaluated. First we evaluate CMCM as a Graph CNN configuration followed by the LSTM. At this point three variants are tested: (i) no extra input and no activation function in the output, (ii) no extra input with a sigmoid activation on the output, (iii) extra inputs (weekend boolean, past day average speed at the same time interval and quarter of the day), (iv) aforementioned extra inputs adding a small LSTM of 5 nodes that evaluates the past hour of the average speed related to the output segment. The fifth variant (v) uses MMCCM as a Graph CNN variant and without extra input.

The model (ii) described above uses a sigmoid activation function on the output. Although the sigmoid activation function is not commonly used in regression problems, we have decided to use it because in traffic data we know the allowed range of predictions, which is between zero and the maximum speed in that segment. Generally, it makes no sense to predict an average speed above the speed limit. Knowing this limitation, the sigmoid function gives the model more room to be more precise compared to a linear (standard) approach. Fig. 6 shows this difference. Most predictions for this segment are between 0.1 and 0.5. Within this area, the sigmoid has a lesser gradient than linear activation. This lesser gradient is what helps the model to be more precise expanding the range on the x-axis from [0.1, 0.5] to [−2.2, 0].

Table 2
Shared vision model overview.

CMCM	Main parameters	Pool	Stride	Activation	Dimensions
Convolution2D	filters = 5	(3,3)	(1,1)	Relu	(5,132,132)
Maxpooling		(4,4)	(2,2)		(5,65,65)
Convolution2D	filters = 15	(3,3)	(1,1)	Relu	(15,63,63)
Maxpooling		(2,2)	(2,2)		(15,31,31)
Flatten					(14415)
Dropout	0,2				(14415)
MMCCM					
Maxpooling		(4,4)	(2,2)		(5,66,66)
Maxpooling		(4,4)	(2,2)		(5,32,32)
Convolution2D	filters = 5	(3,3)	(1,1)	Relu	(5,30,30)
Convolution2D	filters = 5	(3,3)	(1,1)	Relu	(5,28,28)
Maxpooling		(2,2)	(1,1)		(5,14,14)
Flatten					(980)
Dropout	0,2				(980)

Table 3
Hyper-parameters overview defined based on previous experimentation and reference values of literature.

Name	Main parameters	Activation	Size
Input			(5,2,134,134)
Shared vision model	CMCM/MMCCM		5 * (14415)/5 * (980)
Concatenate			(72075)/(4900)
Reshape			(5,14415)/(5,980)
LSTM	n = 250, rdropout = 0,2	Relu	(5,250)
LSTM	n = 250, rdropout = 0,2	Relu	(5,250)
LSTM	n = 250, dropout = 0,2	Relu	(250)
Concat	Optional extra inputs		(254)
Dense	n = 150	Relu	(150)
Dropout	0,2		(150)
Dense	n = 150	Relu	(150)
Dropout	0,2		(150)
Ouput	n = 48	None/Sigmoid	(48)

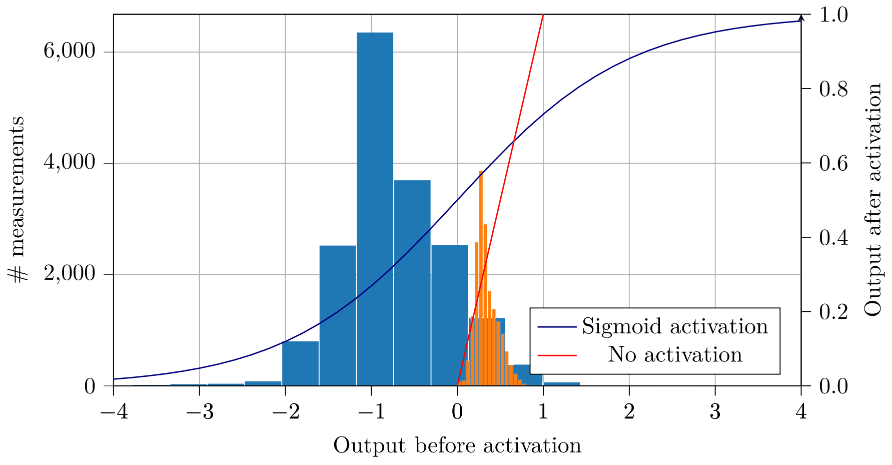

Fig. 6. Histogram of distribution of the normalized output values for segment 2748. The orange histogram refers to the distribution of the normalized output with no activation. The blue histogram refers to the distribution of the normalized output after applying sigmoid function. The red and blue lines show the trends of the linear and sigmoid activation functions, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.3. Experimental set-up

This section presents the experimental set-up used to assess the performance of the data reduction method proposed, the different variations of network architecture designed, the baseline algorithms considered and the performance measures employed for the comparisons.

Regarding the evaluation of the data reduction method proposed, in order to have a baseline to measure its performance, we have compared the simple correlation and temporal correlation variants versus another approach commonly used in literature that consists in choosing direct in- and out-going road links from the target segment, which we will refer to as *In-Outgoing*. As the baseline algorithms, we have used methods commonly found in literature that have shown high performance for TF tasks. Concretely, we have used the following: k-NN, LSTM and SVM. They are optimized using Root Mean Square Propagation with weighted MSE as the loss function. Models are trained for 50 epochs with a callback to the epoch with the best score on the test set. This value was defined based on previous experimentation and the learning curves of the trained models. Moreover, this value of 50 epochs have shown to be sufficient to obtained the best predictions.

The aforementioned methods are also evaluated in terms of the following four measures: Root-Mean-Square-Error (RMSE), Mean-Square Error (MSE), Mean-Absolute Error (MAE) and Mean-Absolute-Percentage Error (MAPE). Their formulations are presented in Eqs. (3), (4), (5) and (6), where r_i and p_i represent the real and predicted values, respectively; w_i represents the weight of the predicted value and s the number of samples. Fig. 7 shows the number of observed cars aggregated over a week. It can be seen that the number of measurements at night is sparse, resulting in untrustworthy data. For this reason, we have considered different weights for different time periods in order to force the model to focus on the time intervals of interest, that is, rush hours. With this idea in mind, we have used the following three variants of the performance indicators:

- Regular: where all the samples in the data have the same weight $w_i = 1$.

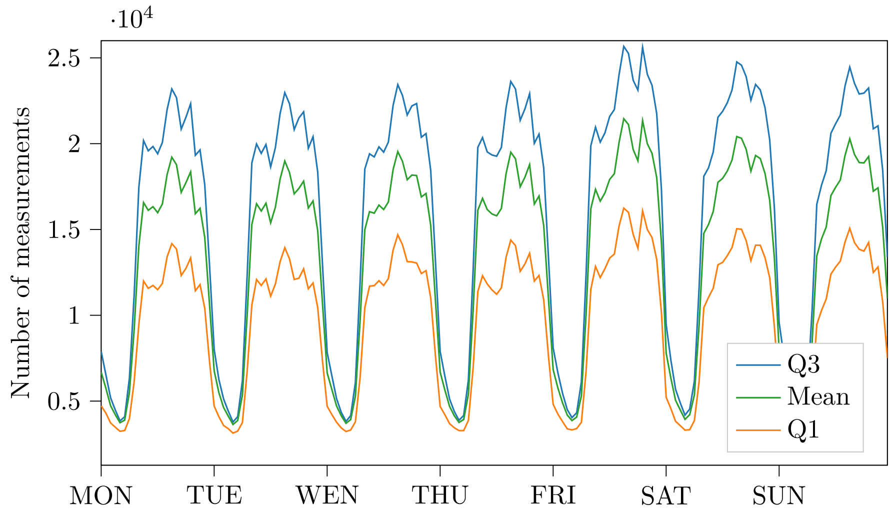

Fig. 7. Number of vehicles over two months, aggregated to one week using summation. Q1 and Q3 represent the first quartile and third quartile.

- Weighted rush-hours: where samples registered outside rush hours (6.00–11.00 and 16.00–21.00) have $w_i = 0.5$ and samples within the rush hour $w_i = 1$.
- Only rush-hours: samples within the rush hours have $w_i = 1$ while samples outside rush-hours are not considered in the calculation and thus have $w_i = 0$.

$$RMSE = \sqrt{\frac{1}{s}\sum_{i=1}^{s} w_i (r_i - p_i)^2} \tag{3}$$

$$MSE = \frac{1}{s}\sum_{i=1}^{s} w_i (r_i - p_i)^2 \tag{4}$$

$$MAE = \frac{1}{s}\sum_{i=1}^{s} w_i \ |(r_i - p_i)| \tag{5}$$

$$MAPE = \frac{100\%}{s}\sum_{i=1}^{s} \frac{w_i \ |(r_i - p_i)|}{p_i} \tag{6}$$

6. Results analysis

This section presents the results analysis of the different experiments performed in this study. Concretely, we show the results of the comparison of the proposed data reduction methods, the comparison among the network architecture variants considered, and the comparison versus the baseline methods. Each comparison will be carried out for both a five-minute prediction and a multi-stage four-hour prediction. With the use of sparse GPS data, there is always noise in the time series coming from probe vehicles. The complete time-series will be considered as ground truth and the Friedman's non-parametric test for multiple comparisons will be used to assess the statistical significance of the results and the Holm post-hoc method to carry out the one-to-many comparisons are used.

Table 4
Performance scoring of pre-processing methods. Errors measured in terms of average speed for (regular/weighted/windowed) situations.

5 min	Ranking	MAE	MAPE	RMSE
In- Outgoing	2.0417	3.481/2.354/1.228	0.126/0.089/0.051	5.110/4.092/2.704
Correlation	2.0833	3.467/2.347/1.226	0.126/0.088/0.051	5.094/4.080/2.699
Time-Correlation	**1.875**	**3.460/2.340/1.220**	**0.125/0.088/0.050**	**5.093/4.076/2.688**
4 h multistep	Ranking	MAE	MAPE	RMSE
In- Outgoing	2.55	4.192/2.977/1.763	0.148/0.107/0.066	5.969/4.995/3.761
Correlation	1.9167	4.135/2.930/1.724	0.146/0.106/0.065	5.912/4.939/3.700
Time-Correlation	**1.5333**	**4.111/2.909/1.708**	**0.145/0.105/0.065**	**5.879/4.905/3.663**

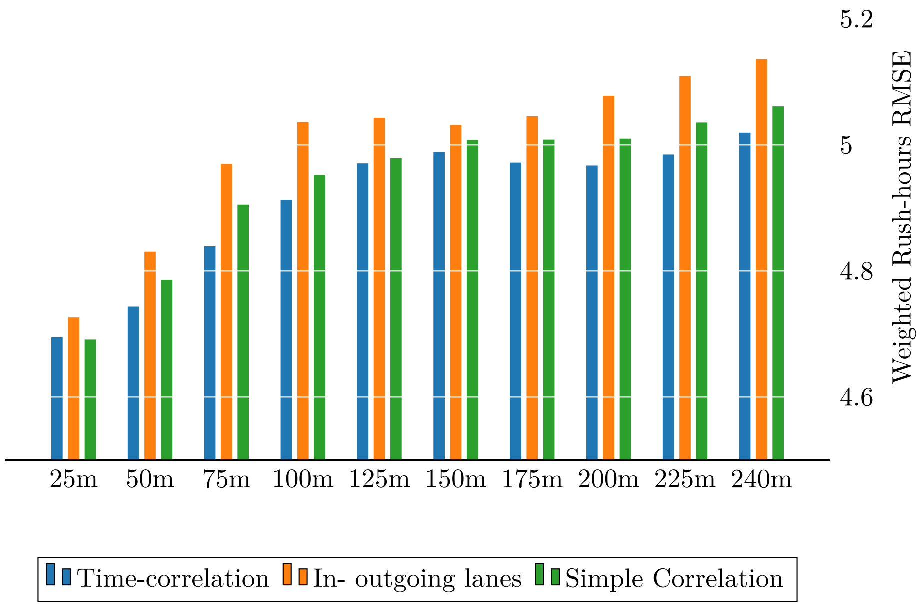

Fig. 8. Average weighted RMSE over all test segments with regards to different pre-processing methods. An aggregation (using the average) is preformed for different prediction horizons. The data used to generate this Figure can be found in the GitHub repository, which contains the supplementary materials of this paper.

6.1. Evaluation of data reduction methods

Three pre-processing approaches were evaluated: In-Outgoing, simple correlation, and temporal correlation. These approaches were tested on all our suggested models to get an overall performance idea. Table 4 shows the average results over segments and time from the aforementioned methods. From these results, we conclude that the spatial features have less influence regarding short-term predictions, meaning that there is no significant difference in performance. The difference can be seen in the four-hour multi-stage prediction. Here the p-values are below 5%, suggesting that the temporal correlation is significantly better compared to regular correlation and in-outgoing traffic approaches. Fig. 8 shows the comparison of these three approaches using the Weighted Rush-hours RMSE for multi-stage long-term prediction in each step. Time-based correlation focuses on overall prediction accuracy and outperforms the other methods in every step except for the first one. We can observe that as the time horizon increases, the difference in performance between Temporal Correlation and the other two methods also increases, which indicates that this method is specially suitable for predictions for long-term time horizons.

To show how the temporal correlation data reduction method helps the model to find spatial and temporal relationships in traffic, in Fig. 9 we show the summation of the correlation of the traffic speed in the last five time intervals and the traffic speed at the segment of interest (highlighted in blue) in different future time horizons for each road link in the city of Xi'an. As can be seen in the scale, green indicates a high correlation in shorter prediction horizons whereas red indicates a high correlation in longer prediction horizons. We observe that road links surrounding the target segments are mostly highlighted green, as they influence its traffic in the near future. As we move away from the segment, the colors start to change to yellow and orange indicating that they are more correlated with intermediate time horizons. And finally, we can also check that the color red is mostly concentrated in the outer segments, showing that the most distant segments have the strongest correlation when the longest time horizons are considered.

6.2. Evaluation of network architecture variants

The models that will be evaluated are presented in Section 5.2. Table 5 presents scores obtained for all the tested models; from the results, we conclude that Convmax-sigmoid outperforms all other models. Convmax, which is in second place, has the exact same hyper-parameters but does not have the sigmoid activation function in the output. This activation function contributes to the accuracy of our model, and thus confirms our hypothesis explained in Section 5. The Holm's post-hoc test shows that there is no significant difference between Convmax and Convmax-Sigmoid. As the models are similar, we will consider the Convmax-Sigmoid to be the best model. Furthermore, these results also show that the influence of proposed contextual information (extra-input) does not help predict in neither short-term nor long-term time horizons of up to 4 h.

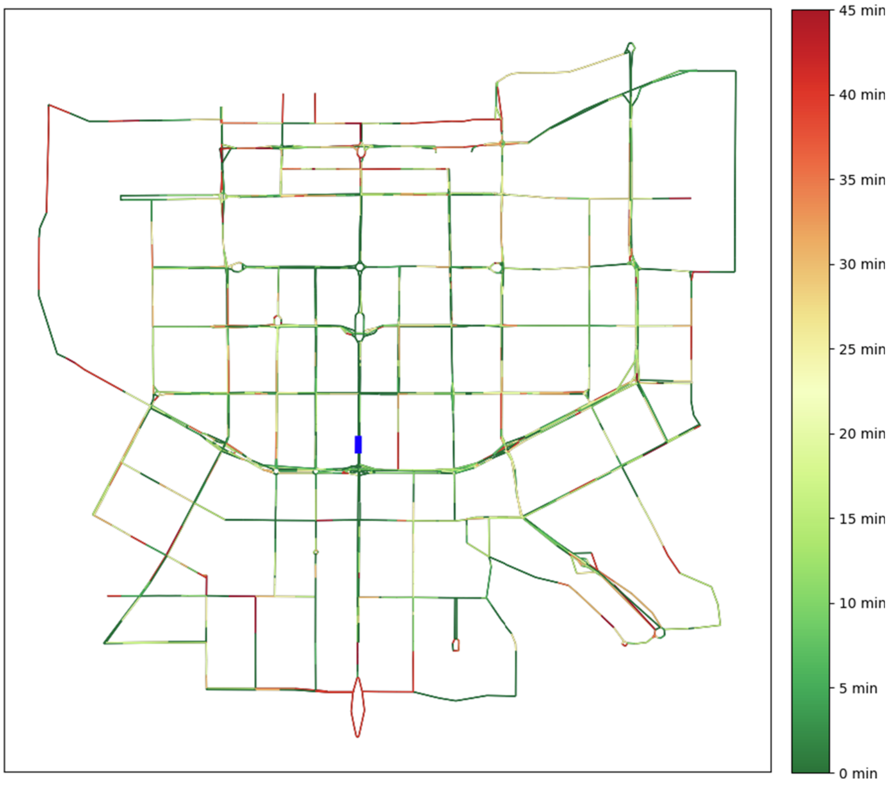

Fig. 9. Summation of the correlation in relation to the last five time steps and the segment of interest (Blue). The maximum correlated time step is illustrated for each segment. For example: 40 min means the last five time steps of this segment is most correlated with the 40 min future prediction of the segment of interest. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 5
Performance scoring of suggested models. Errors measured in terms of average speed for (regular/weighted/windowed) situations.

5 min	Rank	MAE	MAPE	RMSE
Convmax	2.1667	3.433/2.318/1.204	0.124/0.087/0.050	**5.077**/4.062/2.678
Convmax-sigmoid	**1.6667**	**3.430/2.315/1.201**	**0.124/0.087/0.050**	5.078/**4.060/2.667**
Extra-input	3.1667	3.584/2.440/1.295	0.129/0.091/0.053	5.198/4.181/2.811
Extra-input-rnn	4.9167	3.679/2.515/1.350	0.132/0.093/0.055	5.287/4.271/2.913
Maxconv	3.0833	3.527/2.397/1.266	0.127/0.089/0.051	5.148/4.135/2.763
4 h multistep	Rank	MAE	MAPE	RMSE
Convmax	2.0833	4.046/2.847/1.648	0.144/0.104/0.064	5.791/4.808/3.545
Convmax-sigmoid	**1.75**	**4.034/2.839/1.644**	**0.143/0.103/0.064**	**5.783/4.800/3.537**
Extra-input	4.0833	4.201/2.996/1.790	0.146/0.106/0.066	6.004/5.040/3.823
Extra-input-rnn	4.5833	4.215/3.006/1.797	0.147/0.107/0.066	6.012/5.050/3.836
Maxconv	2.5	4.059/2.862/1.664	0.145/0.105/0.065	5.807/4.830/3.578

6.3. Comparison versus baseline algorithms

This section compares the performance of our best network architecture variant versus the baseline algorithms to show its competitiveness. First, our best performing variant Convmax-Sigmoid is compared to the benchmark models using the same data reduction technique. This is followed by a comparison with the benchmark lacking our proposed data reduction approach and using the in -outgoing data reduction method. The benchmark consists of: a LSTM trained with information of the edge itself, a single LSTM, SVM and k-NN trained with the same information contained in adjacency matrices. From Table 6 we conclude that our model outperforms all other benchmarks. The post-hoc analysis showed that all differences were significant except for the LSTM with identical information. There is less of a difference if it is compared to the in-outgoing traffic pre-processing approach, where is applied to the benchmarks. As this is only a five-minute prediction, our pre-processing technique has less influence. The Holm p-values are 15% and 36% for LSTM and LSTM-same-info, respectively, while for the other methods they are below 5%.

Fig. 10 represents the baseline methods with our time-based correlation compared to the Convmax-Sigmoid model using time-

Table 6
Performance scoring of Convmax-Sigmoid compared to benchmark.*indicates in -outgoing pre-processing used. Errors measured in terms of average speed for (regular/weighted/windowed) situations.

5 min	Rank	MAE	MAPE	RMSE
Convmax-sigmoid	**1.417**	**3.430/2.315/1.201**	**0.124/0.087/0.050**	**5.078/4.060/2.667**
k-NN	4.833	3.797/2.592/1.388	0.139/0.098/0.057	5.511/4.462/3.065
LSTM	2.833	3.495/2.363/1.231	0.128/0.089/0.051	5.123/4.104/2.720
LSTM same info	2.25	3.479/2.352/1.226	0.127/0.089/0.051	5.099/4.083/2.702
SVM	3.666	3.653/2.499/1.345	0.132/0.094/0.055	5.239/4.224/2.862
4 h multistep	Rank	MAE	MAPE	RMSE
Convmax-Sigmoid	**1.583**	**3.430/2.315/1.201**	**0.124/0.087/0.050**	**5.078/4.060/2.667**
k-NN*	4.917	3.801/2.601/1.400	0.139/0.098/0.058	5.534/4.483/3.082
LSTM*	2.583	3.496/2.363/1.230	0.129/0.090/0.051	5.124/4.105/2.721
LSTM same info*	2.167	3.486/2.356/1.227	0.128/0.089/0.051	5.104/4.087/2.702
SVM*	3.75	3.672/2.514/1.356	0.133/0.094/0.056	5.243/4.228/2.867

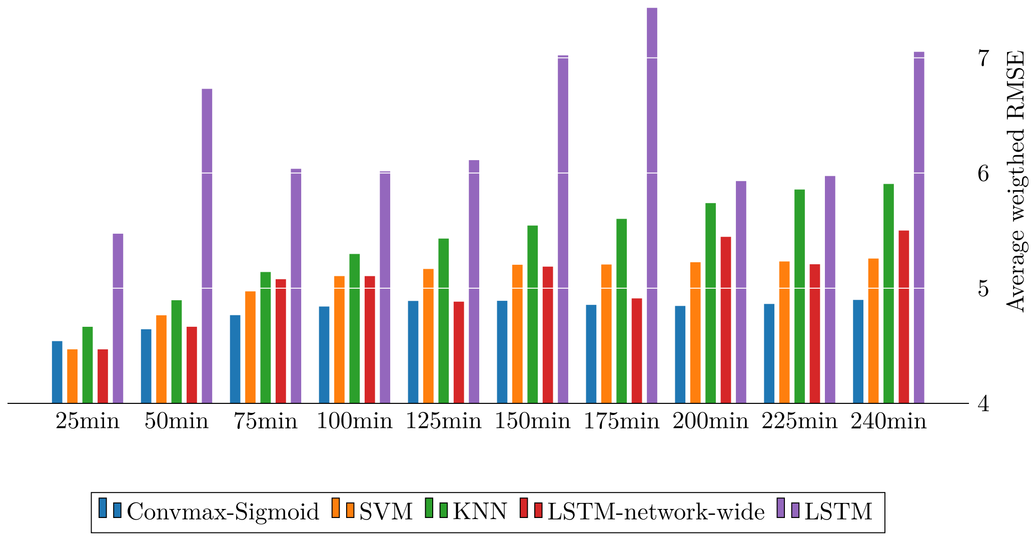

Fig. 10. Average weighted RMSE over all test segments obtained by Convmax-Sigmoid and baseline methods using time-based correlation as pre-processing approach. An aggregation (using the average) is preformed for different prediction horizons. The data used to generate this Figure can be found in the GitHub repository, which contains the supplementary materials of this paper.

based correlation. It can be noted that the exploding gradients cause peaks in the average weighted RMSE of the LSTM methods. These peaks occur due to the recurrent nature of the LSTM architecture. The cells reuse their weights for each iteration, increasing the number of iterations raises the influence of individual weights. These weights can cause exploding or vanishing gradients. Our proposed architecture can prevent this from happening using the FFNN to normalize the influence of each weight. Ignoring these peaks and comparing our model to the benchmarks, it can be concluded that our model outperforms the benchmarks. We can see that the differences between our proposal and the baseline methods increases as the time horizon increases, which suggests that our method becomes more powerful as the complexity of the problem increases. For instance, the difference between our proposal and a simple k-NN at 25mins is 3% whereas this difference increases to 17% at 4 h. With respect to SVM and LSTM-network-wide the differences observed are small although peak differences occur at different time horizons. Another interesting aspect is the robustness of our proposal over the different time horizons the performance is pretty stable. However, we can see that for baseline methods their performance may fluctuate considerably from one time horizon to the next one in some occasions.

Once again, Fig. 11 represents our model using time-based correlation but is now compared to the benchmark using the in-outgoing traffic data reduction approach. The overall performance of the benchmarks decreases and this shows the relevance of our pre-processing method. Once again, we observe that our method becomes more powerful as the prediction horizon increases.

7. Conclusions

Given the relevance of predicting traffic to provide reliable, safer, and greener transportation; TF and ITSs are relevant research

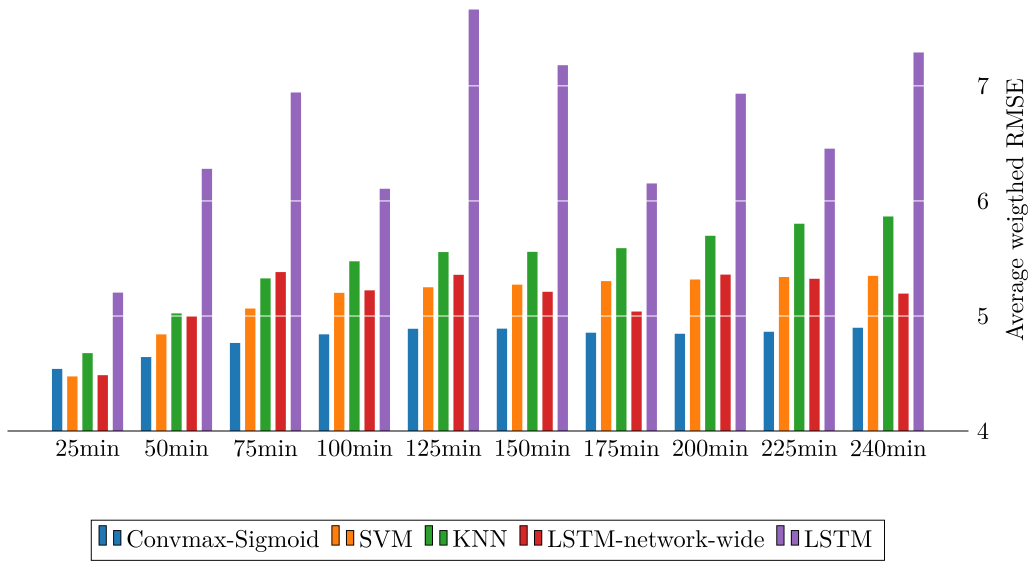

Fig. 11. Average weighted RMSE over all test segments obtained by Convmax-Sigmoid and baseline methods using time-based correlation and in-outgoing pre-processing approaches, respectively. An aggregation (using the average) is preformed for different prediction horizons. The data used to generate this Figure can be found in the GitHub repository, which contains the supplementary materials of this paper.

topics. The availability of spatial and temporal traffic data together with data-driven modelling and simulation techniques have created an opportunity to forecast traffic with high accuracy and efficiency. In this paper, we proposed a hybrid DNN architecture for TF that was able to find both spatial and temporal relationships in traffic data coming from GPS traces. The hybrid DNN consisted of three main components: (i) a Graph CNN, (ii) a LSTM neural network, and (iii) a FFNN. The proposed Graph CNN-LSTM architecture was able to predict traffic in short-term (5 min) time horizons as well as long-term time horizons using multistep predictions (up to 4 h). Furthermore, we have proposed a data reduction technique that selects the n most relevant links from a city-wide road network using temporal-time related correlations. The proposed method was tested over ride-hauling GPS-data from the cities of Xi'an and Chengdu (China) provided by the DiDi Chuxing Gaia Open Data Initiative, and it compared versus state-of-the-art algorithms in TF, such as k-NN, SVM or LSTM.

The main conclusions drawn from the analysis of the results are the following:

- The data reduction technique based on time-related correlation performs better than data reduction methods based only on correlation and upstream and downstream road links.
- The variant of our proposal based on Convmax-sigmoid was the one that obtained the best results in terms of RMSE, MAE and MAPE. However, the difference w.r.t. the Maxconv was not very high and given that this other version has a significantly lower number of parameters, it could be used in some scenarios in which the computational resources for training the model are limited
- The data reduction technique based on time-related correlations also allowed for the improvement of the performance of the baseline algorithms, that is, k-NN, SVM and LSTM.
- Our proposal outperformed baseline algorithms with and without using the data reduction technique in the task of predicting traffic during short-term and long-term time horizons. Furthermore, in long-term predictions, the difference in accuracy between our proposal and the baseline methods became greater as the time horizons increased.

Having discussed the aforementioned contributions and enhancements, the proposed method has improved its performance considerably with respect to the one submitted to the TRANSFOR19 competition. With this new performance, the results obtained by our model would have been the best out of the best five models ranked in the competition in terms of accuracy (RMSE). Further details about this new comparison can be found in the GitHub repository (https://github.com/jsebanaz90/TRC_2019_867-SupplementaryMaterials).

To sum up, we consider this method to be a promising line of research that we aim to keep exploring in the future by extending the predictions from a segment to a network-wide level and increase the prediction time horizon up to 48 h.

CRediT authorship contribution statement

Toon Bogaerts: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Antonio D. Masegosa:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing, Resources, Supervision. **Juan S. Angarita-Zapata:** Investigation, Writing - original draft, Writing - review & editing. **Enrique Onieva:** Writing - original draft, Writing - review & editing, Project administration, Funding acquisition, Supervision, Resources. **Peter Hellinckx:** Funding acquisition, Project administration, Supervision, Resources, Writing - review & editing, Formal analysis.

Acknowledgement

This work was supported in part by the LOGISTAR and MOMENTUM projects, funded by the European Union Horizon 2020 Research and Innovation Programme grant agreements No. 769142 and No. 815069, respectively, and in part by the Marie Sklodoska-Curie Agreement 665959.

References

Aldhyani, T.H.H., Joshi, M.R., 2018. Clustering to enhance network traffic forecasting. In: Mishra, D.K., Nayak, M.K., Joshi, A. (Eds.), Information and Communication Technology for Sustainable Development. Springer, Singapore, pp. 357–364.

Angarita-Zapata, J.S., Masegosa, A.D., Triguero, I., 2019. A taxonomy of traffic forecasting regression problems from a supervised learning perspective. IEEE Access 1–1.

Angarita-Zapata, J.S., Triguero, I., Masegosa, A.D., 2018. A preliminary study on automatic algorithm selection for short-term traffic forecasting. In: Del Ser, J., Osaba, E., Bilbao, M.N., Sanchez-Medina, J.J., Vecchio, M., Yang, X.S. (Eds.), Intelligent Distributed Computing XII. Springer International Publishing, Cham, pp. 204–214.

Atwood, J., Towsley, D., 2016. Diffusion-convolutional neural networks. Advances in Neural Information Processing Systems 1993–2001.

Brownlee, J., 2018. Introduction to Time Series Forecasting with Python, vol 1, first ed., Copyright 2018 Jason Brownlee.

Bruna, J., Zaremba, W., Szlam, A., LeCun, Y., 2013. Spectral networks and locally connected networks on graphs. arXiv 1312.6203.

Cai, P., Wang, Y., Lu, G., Chen, P., Ding, C., Sun, J., 2016. A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting. Transp. Res. Part C: Emerg. Technol. 62, 21–34.

Cui, Z., Henrickson, K., Ke, R., Wang, Y., 2018. Traffic graph convolutional recurrent neural network: a deep learning framework for network-scale traffic learning and forecasting. arXiv 1802.07007.

Cui, Z., Ke, R., Wang, Y., 2018. Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction. CoRR abs/1801.02143.

Defferrard, M., Bresson, X., Vandergheynst, P., 2016. Convolutional neural networks on graphs with fast localized spectral filtering. Adv. Neural Inform. Process. Syst. 3844–3852.

Do, L., Taherifar, N., Vu, L.H., 2018. Survey of neural network-based models for short-term traffic state prediction. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 9, e1285.

Duan, Z., Yang, Y., Zhang, K., Ni, Y., Bajgain, S., 2018. Improved deep hybrid networks for urban traffic flow prediction using trajectory data. IEEE Access 6, 31820–31827.

Ermagun, A., Levinson, D., 2018. Spatiotemporal traffic forecasting: review and proposed directions. Transp. Rev. 38, 786–814.

Forney, G.D., 1973. The viterbi algorithm. Proc. IEEE 61, 268–278.

Polson, G., Sokolov, N.V., 2017. Deep learning for short-term traffic flow prediction. Transp. Res. Part C: Emerg. Technol. 79, 1–17.

Habtemichael, Filmon G., Cetin, Mecit, 2016. Short-term traffic flow rate forecasting based on identifying similar traffic patterns. Transp. Res. Part C: Emerg. Technol. 66, 61–78. https://linkinghub.elsevier.com/retrieve/pii/S0968090X15003186https://doi.org/10.1016/j.trc.2015.08.017.

Hamed, M.M., Al-Masaeid, H.R., Said, Z.M.B., 1995. Short-term prediction of traffic volume in urban arterials. J. Transp. Eng. 121, 249–254.

Howell, S., 2018. Meta-analysis of machine learning approaches to short-term urban traffic prediction. In: Scottish Transport Applications and Research Conference. STAR, pp. 1–15.

Huang, W., Song, G., Hong, H., Xie, K., 2014. Deep architecture for traffic flow prediction: deep belief networks with multitask learning. IEEE Trans. Intell. Transp. Syst. 15, 2191–2201.

Jin, W., Lin, Y., Wu, Z., Wan, H., 2018. Spatio-temporal recurrent convolutional networks for citywide short-term crowd flows prediction. In: Proceedings of the 2nd International Conference on Compute and Data Analysis. ACM, New York, NY, USA, pp. 28–35.

Karimpour, M., Karimpour, A., Kompany, K., Karimpour, A., 2017. Online Traffic Prediction Using Time Series: A Case study. Springer International Publishing, pp. 147–156.

Lana, I., Del Ser, J., Velez, M., Vlahogianni, E.I., 2018. Road traffic forecasting: recent advances and new challenges. IEEE Intell. Transp. Syst. Mag. 10, 93–109.

Li, L., Su, X., Zhang, Y., Lin, Y., Li, Z., 2015. Trend modeling for traffic time series analysis: An integrated study. IEEE Trans. Intell. Transp. Syst. 16, 3430–3439.

Liu, Q., Wang, B., Zhu, Y., 2018. Short-term traffic speed forecasting based on attention convolutional neural network for arterials. Comput.-Aided Civ. Infrastruct. Eng. 33, 999–1016.

Lopez-Garcia, P., Onieva, E., Osaba, E., Masegosa, A.D., Perallos, A., 2016. A hybrid method for short-term traffic congestion forecasting using genetic algorithms and cross entropy. IEEE Trans. Intell. Transp. Syst. 17, 557–569.

Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F., 2015. Traffic flow prediction with big data: a deep learning approach. IEEE Trans. Intell. Transp. Syst. 16, 865–873.

Ma, X., Yu, H., Wang, Y., Wang, Y., 2015. Large-scale transportation network congestion evolution prediction using deep learning theory. PLoS ONE 10, 1–17.

Newson, P., Krumm, J., 2009. Hidden markov map matching through noise and sparseness. In: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, New York, NY, USA, pp. 336–343.

Niepert, M., Ahmed, M., Kutzkov, K., 2016. Learning convolutional neural networks for graphs. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning, vol. 48. pp. 2014–2023 JMLR.org.

Pascanu, R., Mikolov, T., Bengio, Y., 2013. On The Difficulty of Training Recurrent Neural Networks. Technical Report. URL: http://proceedings.mlr.press/v28/pascanu13.pdf.

Pavlyuk, D., 2017. Short-term traffic forecasting using multivariate autoregressive models. Proc. Eng. 178, 57–66.

Shi, Q., Abdel-Aty, M., 2015. Big data applications in real-time traffic operation and safety monitoring and improvement on urban expressways. Transp. Res. Part C: Emerg. Technol. 58, 380–394.

Triguero, I., Figueredo, G.P., Mesgarpour, M., Garibaldi, J.M., John, R.I., 2017. Vehicle incident hot spots identification: an approach for big data. In: IEEE Trustcom/BigDataSE/ICESS, pp. 901–908.

Vlahogianni, E.I., Karlaftis, M.G., Golias, J.C., 2014. Short-term traffic forecasting: where we are and where we're going. Transp. Res. Part C: Emerg. Technol. 43, 3–19.

Yang, S., 2013. On feature selection for traffic congestion prediction. Transp. Res. Part C: Emerg. Technol. 26, 160–169.

Yu, B., Yin, H., Zhu, Z., 2017. Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting. CoRR abs/1709.04875.

Yu, H., Wu, Z., Wang, S., Wang, Y., Ma, X., 2017. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. In: Sensors.

Zhang, J., Zheng, Y., Qi, D., 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence. AAAI Press, pp. 1655–1661.

Zhang, J., Zheng, Y., Qi, D., Li, R., Yi, X., 2016. DNN-based prediction model for spatio-temporal data. In: Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, New York, NY, USA, pp. 92:1–92:4.

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., Li, H., 2018. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. arXiv 1811.05320.