IET Intelligent Transport Systems

Research Article

IET Journals
The Institution of Engineering and Technology

Combining weather condition data to predict traffic flow: a GRU-based deep learning approach

ISSN 1751-956X
Received on 22nd October 2017
Revised 22nd January 2018
Accepted on 1st March 2018
E-First on 27th March 2018
doi: 10.1049/iet-its.2017.0313
www.ietdl.org

Da Zhang[1] ✉, *Mansur R. Kabuka*[1]

[1]*Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL, USA*

✉ *E-mail: zhang.1855@miami.edu*

Abstract: Traffic flow prediction is an essential component of the intelligent transportation management system. This study applies gated recurrent neural network to predict urban traffic flow considering weather conditions. Running results show that, under the review of weather influences, their method improves predictive accuracy and also decreases the prediction error rate. To their best knowledge, this is the first time that traffic flow is predicted in urban freeways in this particular way. This study examines it with respect to extensive weather influence under gated recurrent unit-based deep learning framework.

1 Introduction

Traffic flow prediction acts as an essential component of an intelligent transportation system (ITS). Especially in the urban area, accurate traffic flow prediction provides traveling optimisation for drivers as well as for passengers. Therefore, traffic flow prediction has become one of the most fundamental aspects towards building a smart city in the future. Moreover, due to the dynamic nature of traffic networks, accurately predicting the traffic flow can alleviate traffic congestion propagation, which further cuts down fuel costs and air pollution. Due to the stochastic and non-linear characteristics of traffic flow, previous research exploits artificial neural networks (ANNs) to address this traffic flow prediction problem [1]. However, prominent factors such as weather conditions are typically ignored in previous research as a multisource input. To control traffic condition over past decades, a tremendous number of sensors have been employed in monitoring the traffic situation of freeways. This has generated an enormous amount of high-resolution data. Simultaneously, emerging deep learning framework provides another perspective in mining the data with high dimensionality. Compared to those traditional shallow learning structure, deep learning frameworks can model non-linear phenomena using distributed hierarchical feature representation. This technology draws tremendous attention ranging from speech recognition research to natural language processing field. Understanding how the traffic flow fluctuates under various weather situations can be beneficial for traffic control systems to manage and direct the most appropriate detours efficiently. Therefore, in this paper, taking into account weather data over a particular period, we apply gated recurrent neural network to improve the traffic flow prediction precision.

Previously, several machine learning technologies including parametric and non-parametric techniques have been researched in traffic flow prediction area. In 1970s, Levin *et al.* [2] proposed the autoregressive integrated moving average (ARIMA) model with parameters ARIMA (0,1,1) with the most precise prediction ability. Based on the basic ARIMA model, other variants such as VARIMA [3] model, Kohonen-ARIMA [4] model, and ARIMAX [5][6] model are also developed to improve the traffic flow prediction precision. Parametric techniques are adequate for processing data under regular fluctuations. However, real applications usually involve time-series data without regular variations and are impacted by exterior factors. Under this consideration, non-parametric techniques are introduced to carry out prediction tasks. Those techniques include support vector machine (SVM) [8, 9], feed-forward ANN [10] and decision trees [11] etc.

2 Related work

Traffic flow prediction is an essential component of ITS, and it has been investigated exceedingly during the past decades [12–14]. Time-series analysis has been previously applied to predict traffic flows. Most of the time-series approaches are univariate intrinsically. The univariate time-series models only use historical traffic flow data to predict the future traffic flow. However, in real application, multiple factors need to be considered for drawing more precise prediction. Furthermore, the predictive models can be categorised using different methods. The most prevalent method is to divide the forecasting models into the non-parametric category or parametric category. In this section, we present a short overview of the representative techniques in the two categories, and we illustrate their merit and drawbacks.

2.1 ARIMA

ARIMA method is a parametric technique [15]. A lot of researchers use this technique as the baseline for comparison purpose. Regarding time series data prediction, this ARIMA method uses historical univariate data to analyse the trend and forecast future trend. Therefore, in our paper, we directly extract traffic data sampled hourly without considering influencing factors. Then we apply ARIMA model to fit the data. The common formula of ARIMA is defined as the follows [15, 16]:

$$y_t = \theta_0 + \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{i=1}^{q} \beta_i u_{t-i} + u_t \tag{1}$$

$y_t, y_{t-1}, \ldots, y_{t-p}$ are observed during time periods $t, t-1, \ldots, t-p$, where p is the number of periods considered in the development of the autoregressive term of the model. α_i is the autoregressive parameters. θ_0 is a constant term. q is the number of steps of the moving average process and β_i is the moving average parameters. $u_{t-1}, \ldots, u_{t-q}$ are the random disturbance terms for periods $t-1, t-2, \ldots, t-q$. u_t is the disturbance for period t. Therefore, 'AR' represents the autoregressive component of the model which involves regression on historical values. 'MA' refers to the moving average component of the model. For instance, if an ARIMA model is given by ARIMA(p,d,q), it means that this model requires d order of differencing, p steps of autoregressive process, and q steps of moving average component.

2.2 ARIMAX

ARIMA is efficient and simple to implement. However, real applications which apply ARIMA method are usually influenced by external factors. For instance, the traffic accidents occurrence is largely impacted by weather factors especially under extreme weather conditions. These influencing factors include both time-series and independent factors such as weather data. Based on these factors, ARIMA technique was extended as a multivariate variant ARIMAX [17] to compensate for the ARIMA model described as follows:

$$y_t = \theta_0 + \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{i=1}^{q} \beta_i u_{t-i} + u_t + \sum_{k=1}^{N} \beta_k Y(t,k) \tag{2}$$

Comparing to (1) in the ARIMA model, the additional term $\sum_{k=1}^{N} \beta_k Y(t,k)$ in formula (2) is defined as follows: $Y(t,k)$ is the external variable matrix in which each column represents a sample of time-series data, β_k represents the external variable parameters, and N is the number of influential factors. Therefore, the previous ARIMA(p,d,q) model evolves to a more advanced model ARIMAX(p,d,q) that has autoregressive steps p, differencing order d, moving average q and exterior influencing factors. In this way, the ARIMA model expands to the ARIMAX model when taking exterior factors.

2.3 Support vector regression

Support vector regression (SVR) is a non-parametric method for time-series prediction tasks. It is the most prominent application of the SVM [7] techniques. The basic idea of SVR [18] is to map the data in low-dimensional space $\boldsymbol{x}$ into high-dimensional space $\boldsymbol{F}$ using the non-linear transformation method Φ. Then, we perform linear regression within the high-dimensional space. The process is described as follows:

$$f(\boldsymbol{x}) = (\boldsymbol{w} \cdot \Phi(\boldsymbol{x})) + b, \quad \text{with } \Phi: R^n \to \boldsymbol{F}, \boldsymbol{w} \in \boldsymbol{F} \tag{3}$$

Here, w and $\Phi(\boldsymbol{x})$ are computed iteratively in the high-dimensional space $\boldsymbol{F}$. By minimising the empirical risk of f and complexity [19] degree $||w||^2$, this process can be written as a convex optimisation procedure as shown in formula (4). SVR is especially useful when the underlying function that maps input variables to output results is unknown, or the mathematical expression between input variables to output data is too expensive to be obtained due to the large number of external influencing factors [20]

$$\begin{aligned} &\text{minimise} \quad \frac{1}{2}||w||^2 \\ &\text{subject to} \quad |y_i - (w \cdot \Phi(x_i)) - b| < \epsilon \end{aligned} \tag{4}$$

2.4 Random forest regression

Another ideal non-parametric technique for traffic flow prediction is random forest technique. This technique originated from the regression tree [11]. Random forest consists of multiple regression trees. A typical regression tree is comparable to a classification tree. However, the target variable takes ordered values. A regression model is then applied to each tree node to derive the predicted value while minimising the loss entropy [11]. During the tree construction process [21], three steps are involved: (i) choosing splitting rules; (ii) setting criterion to terminate splitting and stop at a leaf node; (iii) assigning a target value y to each leaf node. When the internal node is split, we choose and compare a variable v_i with a threshold. Then we split the node according to this binary value result. After the splitting process, we calculate the prediction error in formula (5) for each node and apply the regression model at each node iteratively [21]

$$E(t) = \frac{1}{N_t} \sum_{i=1}^{N_t} (y_{t_i} - \hat{y}_t)^2 \tag{5}$$

When we perform traffic volume prediction tasks using this model, for each node t, N_t is the size of samples in the node and y_{ti} is the traffic flow of the i_{th} data falling into node t. $\hat{y}$ is the average of the traffic flow of the samples falling into node t. The best set x^* consisting of m variables and threshold a^* that split node t are selected to minimise the error $E(t)$ in formula (6). For any split of node t, the left child and right child are defined as t_L and t_R. Their error rate is defined as $E(t_L)$ and $E(t_R)$. Thus, the decrease of $E(t)$ is calculated as

$$\Delta(t) = E(t) - E(t_L) - E(t_R) \tag{6}$$

Due to the deficiency of the high variance of regression tree, we construct a bag of trees as a random forest to leverage the variance [22, 23]. Assuming we set B as the number of trees in the forest, ρ as the pairwise correlation parameters with variance σ^2, the average variance is leveraged using the following formula (7) by randomly selecting the splitting variables at each node

$$\text{Var}(\mu) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \tag{7}$$

3 Methodology

3.1 Problem definition

Previously, traffic flow prediction problem can be defined as follows [24]: X_i^t denotes the observed number of vehicles passing through an active detector on a freeway during the tth time interval at the ith observation location in a transportation network. Given a sequence $\{X_i^t\}$ of observed traffic volume data, $i = 1, 2, \ldots, m$, $t = 1, 2, \ldots, T$, the problem is to forecast the traffic volume at time interval $(t + \delta)$ for certain future time steps δ [24].

3.2 Recurrent neural network architecture

A typical recurrent neural network (RNN) can be described as a deep generative architecture as shown in Fig. 1*a*. It has one input layer $\boldsymbol{x}$, a hidden layer $\boldsymbol{h}$, and an output layer $\boldsymbol{o}$. It can be regarded as a straightforward adoption of the standard feed-forward neural network that models data with time stamps [25]. As shown in Fig. 1*a*, the input layer of RNN receives an input vector and then forward it to the hidden layer and finally predict a set of data in future time steps at the output layer. The RNN's high-dimensional hidden states provide it with a significant expressive power to model non-linear evolution. It enables the hidden states of the RNN to integrate information over many time steps and use it to make accurate predictions. RNN is particularly useful for modelling the sequential data in text and speech recognition fields [25]. Additionally, a lot of research has applied this RNN method to model traffic data which embodies temporal information [26, 36]. Thus, the traffic flow prediction problem defined in Section 3.1 can be represented as

$$\boldsymbol{x} = (x_1, x_2, \ldots, x_T) \tag{8}$$

$$\boldsymbol{o} = (o_1, o_2, \ldots, o_\delta) \tag{9}$$

$$\boldsymbol{h}^n = (h_1^n, h_2^n, \ldots, h_T^n) \tag{10}$$

$$P(x_{t+1} = y_j | x_t, \ldots, x_1) = y_{t,j} \tag{11}$$

Here, we define the problem more specifically under the scenario of simplest RNN: given a training sequence of traffic flow $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$ from time 1 to T, the RNN model learns to predict the next δ value $\boldsymbol{o} = (o_1, o_2, \ldots, o_\delta)$ from T of its input variables. The hidden layer $(h_1^n, h_2^n, \ldots, h_T^n)$ not only takes the current input $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$ but also takes into account previous states they perceive, which is plotted as an arrow pointed out from previous neuron to the next neuron in Fig. 1*a*. The circle implies that RNN performs the same task for every element of a sequence,

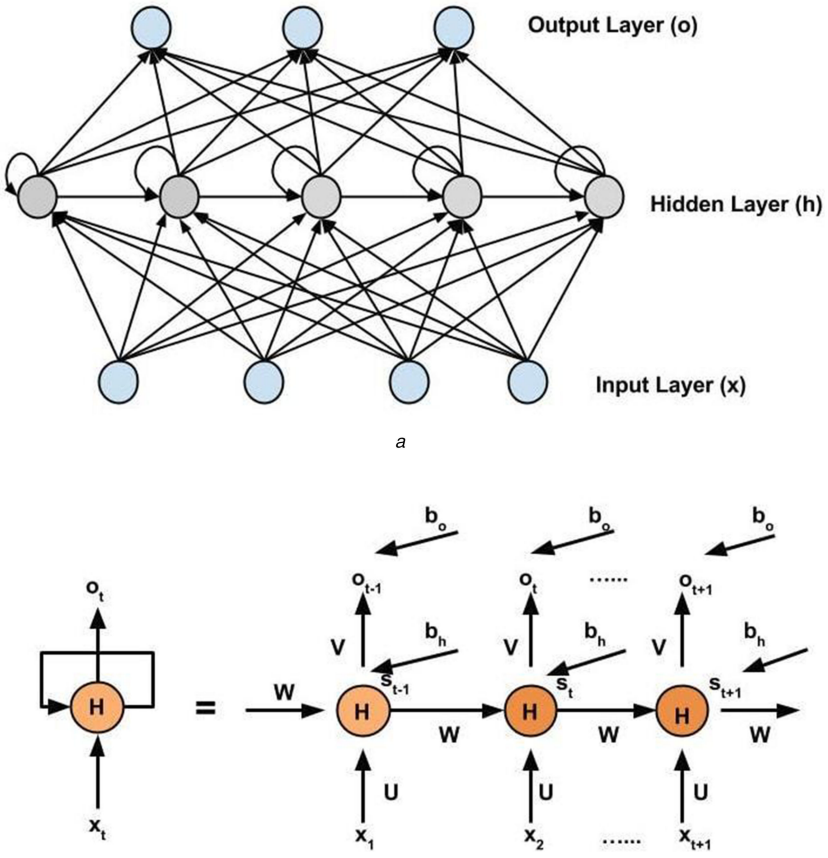

Fig. 1 *RNN architecture*
(a) Typical RNN architecture, ***(b)*** Unfolded RNN

with the output being depended on the previous computations. The sequential information is captured in the hidden states. It manages to propagate certain time steps as it avalanches forward to influence the new forthcoming examples. In a traditional neural network, we assume that all inputs (and outputs) are independent of each other. However, many tasks have a dependent relationship between current states and previous steps in a real application such as traffic flow prediction sampled by time stamps. Fig. 1*b* depicts the unfolded internal RNN architecture. Equations (12) and (13) are the mathematical representation of this unfolded process.

For time $\boldsymbol{t}$ from **1** to $\boldsymbol{T}$

$$\boldsymbol{s}_{t+1} = \tanh(\boldsymbol{U}_{xh}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{s}_t + \boldsymbol{b}_h) \tag{12}$$

$$\boldsymbol{o}_{t+1} = \mathrm{softmax}(\boldsymbol{V}_{ho}\boldsymbol{s}_{t+1} + \boldsymbol{b}_o) \tag{13}$$

Here, $\boldsymbol{U}_{xh}$ represents the weights matrix from the input layer to hidden layer, $\boldsymbol{W}_{hh}$ represents the duplicated recurrent weight matrix, and $\boldsymbol{V}_{ho}$ is the hidden to output weight matrix. b_{h} and b_{o} are the biases added to the hidden and output layer. RNN can be unfolded over time, each feedback loop will be expanded as single layer feed-forward neural network at each time stamp in Fig. 1*b*. In this case, RNN can be efficiently trained using the well-known back prorogation through time (BPTT) algorithm. However, BPTT has difficulty training the sequence with long-term dependency since it can be converted into a deep architecture with multiple layers, and results in vanishing gradient as well as exploding gradient issues.

3.3 Gated recurrent unit

Previously, long short-term memory (LSTM) [28] described in Fig. 2 is pervasively used as a gating mechanism in RNN to solve the vanishing and exploding gradient problems in BPTT algorithm. Based on LSTM gating mechanism, gated recurrent unit (GRU) shown in Fig. 3 was first proposed by Cho *et al.* [29]. GRU models the data flow inside the unit similar to LSTM. However, different from LSTM, GRU does not have separate memory cells, which makes GRU more computationally powerful when training the data. GRU and LSTM have been utilized in recent researches to capture the non-linear traffic behavior efficiently [30] [31] [32]. As depicted in Fig. 3, the input and forget gates in LSTM [28] are coupled by an update gate z, and the reset gate r is applied directly to the previously hidden state. Thus, the responsibility of the reset gate in an LSTM unit is split up into both r and z.

The reset gate $\boldsymbol{r}$ determines if we need to combine the current state with previous information, while the update gate $\boldsymbol{z}$ defines how much of the memory to keep around. A Sigmoid gate squashes numbers between 0 and 1, describing how much information should be passed through. Here, 1 means letting all the information go through the cell states. 0 means no information is allowed to pass the cell states. tanh gates are used to put the value between −1 and 1. The GRU cell is computed as follows:

$$z = \sigma(x_t\boldsymbol{U}^z + \boldsymbol{s}_{t-1}\boldsymbol{W}^z) \tag{14}$$

$$r = \sigma(\boldsymbol{x}_t\boldsymbol{U}^r + \boldsymbol{s}_{t-1}\boldsymbol{W}^r) \tag{15}$$

$$\boldsymbol{h} = \tanh(\boldsymbol{x}_t\boldsymbol{U}^h + (\boldsymbol{s}_{t-1} \circ \boldsymbol{r})\boldsymbol{W}^h) \tag{16}$$

$$\boldsymbol{s}_t = (1 - z) \circ \boldsymbol{h} + z \circ \boldsymbol{s}_{t-1} \tag{17}$$

3.4 Traffic flow prediction with multiple sequential inputs

Predicting the traffic flow is crucial for an ITS. Meanwhile, analysing the critical factors that influence the traffic flow can provide more accurate predictions in the future. Since various features of weather data are also time-series data in every time step, instead of predicting traffic flow merely from previous traffic flow information, we embed weather features into traffic flow data.

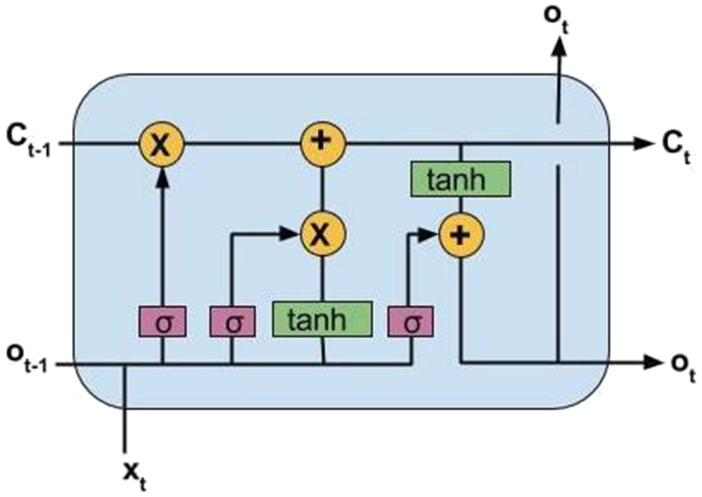

Fig. 2 *Long- and short-term memory cell*

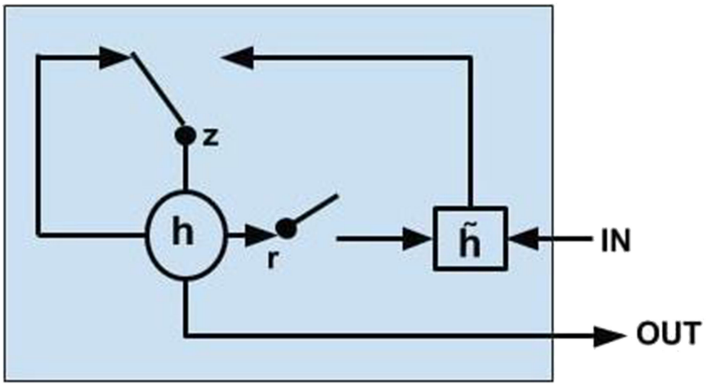

Fig. 3 *Gated recurrent unit cell*

$$X = \begin{matrix} \begin{matrix} m_{11} & m_{12} & m_{13} & \cdots & m_{1\tau} \\ m_{21} & m_{22} & m_{23} & \cdots & m_{2\tau} \end{matrix} \\ \begin{matrix} m_{11} & m_{12} & m_{13} & \cdots & m_{1\tau} \\ m_{21} & m_{22} & m_{23} & \cdots & m_{2\tau} \end{matrix} \\ \begin{matrix} m_{11} & m_{12} & m_{13} & \cdots & m_{1\tau} \\ m_{21} & m_{22} & m_{23} & \cdots & m_{2\tau} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{D1} & m_{D2} & m_{D3} & \cdots & m_{D\tau} \end{matrix} \end{matrix} \tag{19}$$

Thus, at each time step, we use a vector consisting of weather features and traffic time stamps to represent the highly dimensional vector space $\mathbb{R}^D$. We rewrite the RNN formulas using the following representation:

$$\boldsymbol{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & \cdots & m_{1\tau} \\ m_{21} & m_{22} & m_{23} & \cdots & m_{2\tau} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{D1} & m_{D2} & m_{D3} & \cdots & m_{D\tau} \end{bmatrix} \tag{18}$$

Here, the input data $\boldsymbol{X} \in \mathbb{R}^{S\times\tau\times D}$ is an $S \times \tau \times D$ dimension tensor comprising of layered matrices $\boldsymbol{M} \in \mathbb{R}^{D\times\tau}$. Each column $\boldsymbol{m_i}$ in $\boldsymbol{M}$ is a D dimension vector represented by $\boldsymbol{m_i} = (m_{1i}, \ldots, m_{Di}) \in \mathbb{R}^D$. Dimension D is determined by the number of features from the weather data set combined with sampled traffic data. τ in $\boldsymbol{M}$ is the number of time steps we refer to when predicting σ steps output data. S is the number of samples we use when constructing the tensor. Therefore, the shape of the input data in our paper is a three-dimensional tensor with the shape $S \times \tau \times D$. The size of the hidden matrix $\boldsymbol{W}_{hh}$ in formula (12) is determined by the number of neurons in the hidden layer while the input matrix $\boldsymbol{U}_{xh}$ is determined by the hidden neurons and the input feature shape. Thus, if we have N neurons in the hidden layer, the size of $\boldsymbol{W}_{hh}$ is necessary to be $N \times N$ and the size of $\boldsymbol{U}_{\boldsymbol{xh}}$ is $N \times D$ with random initial values.

3.5 Deep GRU recurrent neural networks model

Based on the above discussion, we propose our deep GRU recurrent neural network (DGRNN) model described in Fig. 4 with deep GRU hidden layers. In our model, we replace neurons in the hidden layer by GRU units to discover the correlation within time series data in both short- and long-time steps. The final prediction layer in our model is implemented with a fully connected layer connected to the output layer. As shown in Fig. 4, each neuron will contribute partial information to the output result during feed-forward process and the parameter matrix for each layer of hidden neurons will get updated during back propagation process.

4 Experiment setting

This section describes the experimental data set, the system configuration of the machine and the optimisation function during the learning phase. It also describes the performance evaluation criteria and activation function used when conducting the experiments. In the experiments, we integrated weather condition data and use 100 h of historical data to predict 12 h future traffic flow. Our model was trained with BPTT [33] method and Adaptive Moment Estimation (Adam) algorithms using Tensorflow back end. Google's TensorFlow [34] is an open-source framework for

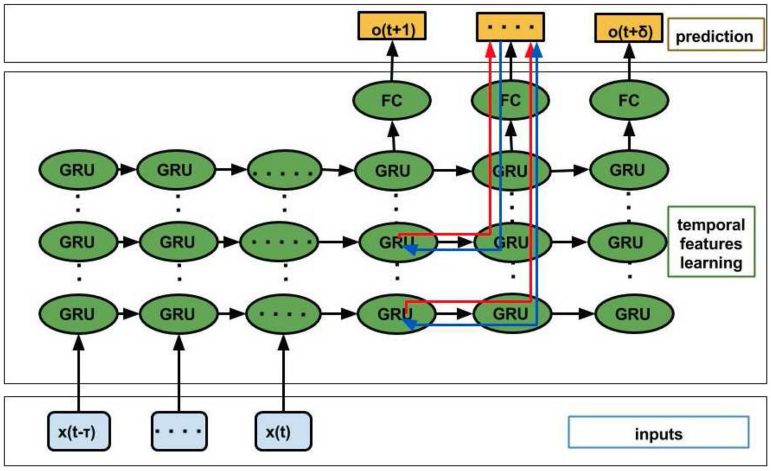

Fig. 4 *DGRNN model*

Table 1 Multiple input features

Parameter description	Abbreviation
average weed speed	AWND
precipitation	PRCP
maximal temperature	TMAX
minimal temperature	TMIN
ice Fog (1 true, 0 false)	WT01
freezing Fog (1 true, 0 false)	WT02
smoke (1 true, 0 false)	WT08
the day is weekend (1 true, 0 false)	is_Holiday
time stamp based on 24 h per day	time_num
day stamp based on 7 days per week	weekday_num
vehicle miles travelled	VMT

Table 2 System configuration

operating system	Ubuntu 16.04
GPU version	NVIDIA Tesla K80
GPU memory	12 GB
python version	Python 3.5.2
TensorFlow	Version 1.0.1

accessing the state-of-the-art machine learning and deep learning algorithms. It has been used for conducting research and deploying deep learning systems into production in the applications in multiple research areas. Based on TensorFlow, we use Keras library [35] to build and train our model. We trained network using adaptive moment estimation (Adam) method [36] since this method adapts its learning rate automatically and thus has better performance than other optimisation methods.

4.1 Dataset description and system configuration

The dataset we use in this paper is extracted from Caltrans performance measurement system (PeMS), which includes the traffic data collected in real time from over 39,000 individual detectors in the state of California. These sensors span the freeway system across all the main metropolitan areas of the State of California [37]. We analysed the data sampled hourly extracted from the PeMS website [37] and combined it with weather feature data from the National Oceanic and Atmospheric Administration (NOAA) repository [38]. After combining the features, the complete feature array is listed in Table 1. As listed in Table 1, feature vehicle miles traveled (VMT) indicates vehicle miles travelled for the specific freeway sampled hourly. The value of this feature is provided by PeMS website and is calculated by multiplying the volume of hourly traffic on a freeway segment by the length of the road. Therefore, VMT summarises all the segments' traffic flow to give the user a total traffic flow information for the geographical area of concern. At the same time, we combined the weather data collected from stations covering the area of concern of the same period from NOAA repository [38]. The data features are weather types (WT01–08), precipitation (PRCP), average wind speed (AWND), minimal temperature (TMIN) and maximal temperature (TMAX). The weather data sampled hourly aligned with traffic flow data serve as multi-sequential inputs to our proposed DGRNN model. In order to prove the robustness of our proposed model, we apply our proposed model on three datasets. The first dataset is 2-month VMT data from 1 November 2016 to 31 December 2016 of the SR237-W freeway located in Santa Clara county. The second dataset is from US101-N located in same area during the same period. Then, we extend the time period to 2 years and extract the dataset from US101-N again to test our model. We extend the time period to 2 years such that the influence of seasonal features is taken into account while predicting the traffic flow.

The PeMS [37] only provides the traffic volume data sampled by hour for each freeway. Therefore, we keep the data integrity without slicing it into smaller time intervals. Before applying DGRNN model on the data, we first need to perform feature scaling on both the weather condition data and traffic flow data so that the data are standardised in the range of $[0, \alpha]$ as shown in formula (20). Here, α is a normalising factor. For simplicity, we initialised $\alpha = 1$ such that the data is normalised in the range of [0, 1]. Moreover, we split the dataset into two parts. We used 10% as validation data and 90% as training data

$$S = \alpha \times \frac{x - \min(x)}{[\max(x) - \min(x)]} \tag{20}$$

In Table 2, we listed the system configuration. We created an instance and ran experiments on Google Compute Engine virtual machine. The virtual instance we created has eight virtual CPUs and 52 GB memory. Additionally, we run our DGRNN model on the graphic processing unit (GPU) which has highly parallel structure. Thus, it is more effective than general-purpose CPUs for algorithms as the processing of large blocks of data is done in parallel. font = scriptsize

During the experiments, we use three performance evaluation criteria to evaluate our model's prediction accuracy, which is defined as the following formula. Here $\hat{f}_i$ represents predicted traffic flow and f_i denotes actual traffic flow.

Mean absolute error (MAE) [39]

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |f_i - \hat{f}_i| \tag{21}$$

Mean-squared error (MSE) [40]

Table 3 Comparison of running results

Model	Evaluation criteria	Error rate
two hidden layer simple RNN (50 hidden units for each layer)	MSE	0.0014
	MAE	0.024
	RMSE	0.037
two hidden layer DGRNN (50 hidden units for each layer)	MSE	7.45×10^{-4}
	MAE	0.019
	RMSE	0.027
two hidden layer DGRNN (500 hidden units for each layer)	**MSE**	$\mathbf{3.76 \times 10^{-5}}$
	MAE	**0.0079**
	RMSE	**0.0019**
two hidden layer LSTM (500 hidden units for each layer)	MSE	8.72×10^{-5}
	MAE	0.0090
	RMSE	0.0093
three hidden layer DGRNN (500 hidden units for each layer)	MSE	0.00044
	MAE	0.0082
	RMSE	0.02
ARIMA (0,1,1)	MSE	0.007
	MAE	0.058
	RMSE	0.083
support vector regression	MSE	0.003
	MAE	0.048
	RMSE	0.058
random forest regression	MSE	0.002
	MAE	0.029
	RMSE	0.042

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(f_i - \hat{f}_i)^2 \tag{22}$$

Root mean-squared error (RMSE) [40]

$$\text{RMSE} = \left[\frac{1}{n}\sum_{i=1}^{n}(|f_i - \hat{f}_i|)^2\right]^{1/2} \tag{23}$$

Historically, a common choice of activation function for each hidden neuron is among Sigmoid, tanh and rectified linear unit (ReLU) functions. For accuracy and efficiency, in our paper, we use ReLU activation function defined as following formula as our activation function:

$$\text{ReLu}: f(x) = \max(0, x) \tag{24}$$

4.2 Running results

In the experiments, we first compare our proposed DGRNN model with classical machine learning methods described in Section 2. In our paper, we implemented the classical machine learning techniques using Sci-kit python packages [41]. Then we compare our DGRNN model with the SVR, random forest regression, and ARIMA models separately using the 2-month SR237-W dataset. The running results are listed in Table 3. Among these classical models, we used the ARIMA model as the baseline without taking into account weather feature data. From Table 3, we can see that our proposed DGRNN model for the short-term prediction of the traffic flow outperforms classical machine learning algorithms including ARIMA, SVR, and random forest regression using all three evaluation metrics. MSE, RMSE, MAE. The bold value in Table 3 presents our DGRNN model has smallest error rate among the comparing methods.

Then we compare our DGRNN model with various recurrent neural networks (RNN) models using the same 2-month SR237-W traffic flow dataset. We trained the DGRNN model with empirical predictors for 300 epochs and evaluated the error rate using three evaluation metrics. This time, we use basic RNN as the baseline for comparison. We started from two hidden layers with 50 hidden units in each layer and run 300 epochs. The maximum average prediction accuracy improvement of the DGRNN model is up to 6.8% compared with the baseline RNN as plotted in Fig. 5. Then, we increase the number of hidden neurons in each layer to 500 and compare the performance of DGRNN model with LSTM-based RNN. From the results in Table 3 and results plotted in Fig. 5, we can see that DGRNN performs better than LSTM RNN for performing long-term prediction in the future. Additionally, since GRU has less number of gates, it needs less parameters in each unit, which makes the training time less than LSTM RNN. After that, we further increase the number of layers. However, the performance is not improved. Rather than that the prediction accuracy dropped significantly after the 200 epochs. This phenomenon is caused by the over-fitting problem when the neural network gets deeper. Therefore, we choose the best DGRNN model having highest prediction accuracy and lowest error rate. It has two hidden layers with 500 hidden units in each layer.

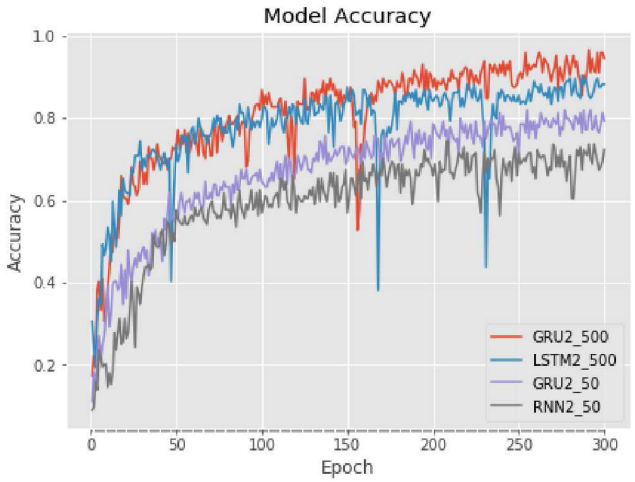

Fig. 5 *Comparison of SR237-W traffic flow prediction accuracy*

Finally, we use the best trained DGRNN model to predict the traffic flow and plotted the model accuracy for both with and without weather condition data for three dataset mentioned in

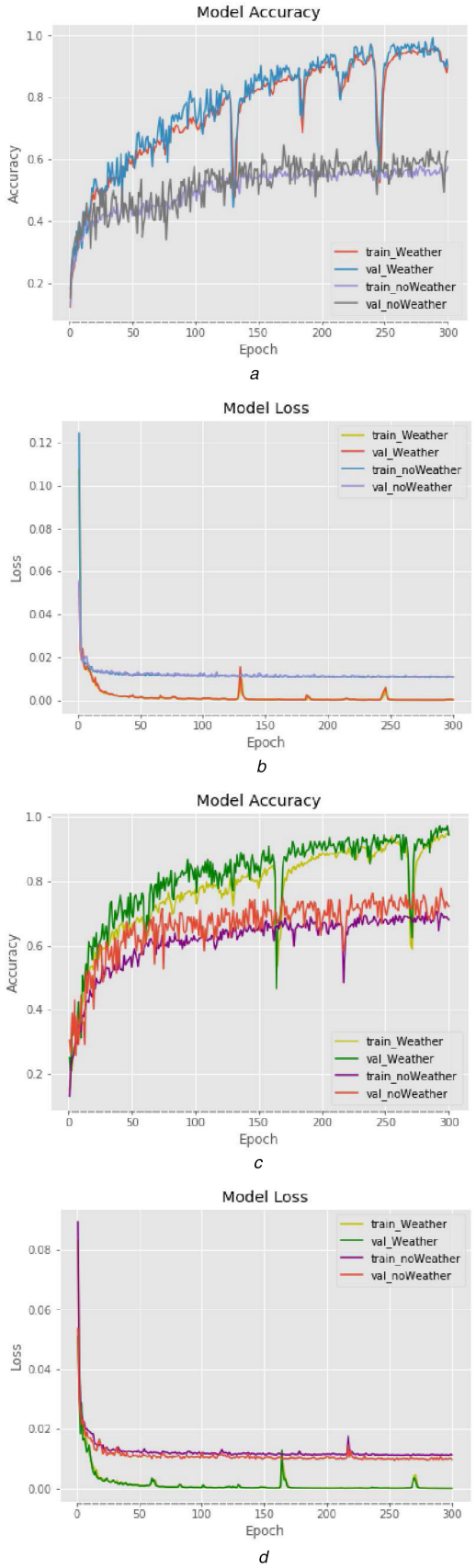

Fig. 6 *Two-month US101-N and SR237-W VMT prediction accuracy and MSE loss rate*
(a) US101-N VMT prediction accuracy, ***(b)*** US101-N VMT prediction MSE loss rate, ***(c)*** SR237-W VMT prediction accuracy, ***(d)*** SR237-W VMT prediction MSE loss rate

Section 4.1. We plotted their prediction accuracy separately in Figs. 6 and 7. From Figs. 6*a*, *c* and Fig. 7*a*, we can see that the average accuracy of the DGRNN model is over 94% for all the prediction tasks. In Figs. 6*b*, *d* and Fig. 7*b*, we use the same model

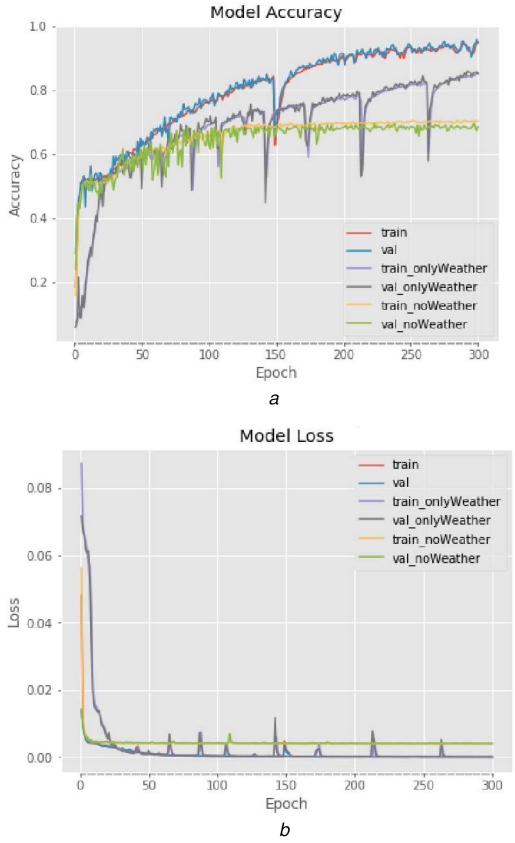

Fig. 7 *Two-year US101-N VMT prediction accuracy and MSE loss rate*
(a) Prediction accuracy, ***(b)*** Prediction MSE loss rate

to plot the MSE loss rate again with and without weather condition data. Since the data has been regularised in the range of [0, 1], the loss rate was calculated using preprocessed data. From the comparison results, we deduced that combining with weather condition data, the prediction accuracy is promoted and the prediction loss is decreased. Figs. 8*a* and *b* present a visualisation tendency of part of the validation dataset of 2-month SR237-W traffic flow data using the proposed DGRNN model both with and without weather condition data. From the results, it is clear that combing weather features to traffic flow data makes the model fit actual traffic flow data much better. It is also worth noting that, when we use 2-year dataset as shown in Figs. 7*a* and *b*, our DGRNN model performs more stable than 2-month dataset. Additionally, the seasonal weather data performs as a more influential factor as the increment of prediction accuracy is larger than small dataset. As can be seen in Fig. 7*a*, the prediction accuracy ignoring weather data stays unchanged after it achieves 70%. The loss rate stays the same after around 50th epochs without further decreasing. This prediction accuracy was steady and commensurated with other reported results [42]. Furthermore, if we eliminate the features of {is_holiday, time_num and weekday_num}, the prediction accuracy is still improved but not as good as accuracy with complete features. Finally, the prediction accuracy achieves the highest score with complete feature set as shown in Figs. 7*a* and *b*.

5 Conclusion

In our paper, we propose the DGRNN model to improve the long-term traffic flow prediction accuracy while taking into account weather conditions. Our paper focuses on traffic flow prediction, but actual weather feature data which includes precipitation, average wind, maximal temperature, minimum temperature and complete weather types are incorporated into our models. We first compare our DGRNN model with both parametric and non-parametric classical machine learning techniques including ARIMA, SVR and random forest regression model. The running results show that our model achieves the least error rate using three

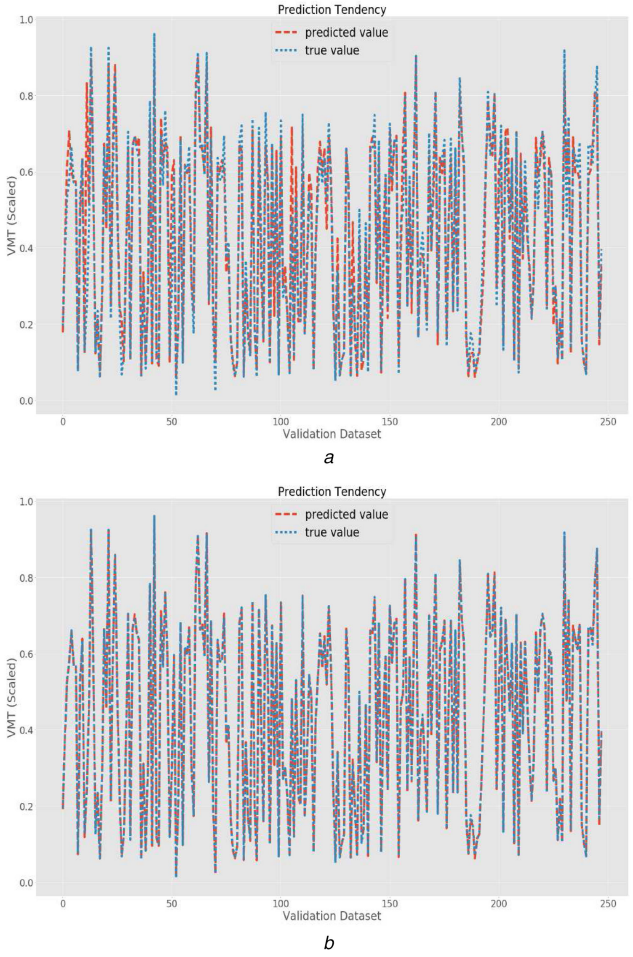

Fig. 8 *Two-month SR237-W VMT forecast result and the actual data*
(a) Without weather feature data, ***(b)*** With weather feature data

evaluation criteria: MSE, MAE and RMSE. Additionally, we compare our model with simplified RNN and LSTM RNN models, which are prevalent used deep learning framework to predict traffic flow data. Comparing with these models with the same hyperparameters, our model has prediction accuracy improvement up to 5.8%. Also, the MSE error achieves the lowest among all the models. Then we compare the performance using DGRNN model both with and without taking into weather condition data. Taking into weather condition data as multiple sequential inputs, our model improves traffic flow prediction accuracy up to 25% compared with models without taking into weather condition data. Therefore, we believe that our paper could have useful applications in the areas of modelling, analysing and mining big traffic data.

6 References

[1] Smith, L.B., Demetsky, M.J.: 'Short-term traffic flow prediction: neural network approach', *Transp. Res. Rec.*, 1994, **1453**, pp. 98–104

[2] Levin, M., Tsao, Y.D.: 'On forecasting freeway occupancies and volumes (abridgment)', *Transp. Res. Rec.*, 1980, **773**, pp. 47–49

[3] Kamarianakis, Y., Prastacos, P.: 'Forecasting traffic flow conditions in an urban network: comparison of multivariate and univariate approaches', *Transp. Res. Rec.*, 2003, (1857), pp. 74–84.

[4] Van Der Voort, M., Dougherty, M., Watson, S.: 'Combining Kohonen maps with ARIMA time series models to forecast traffic flow', *Transp. Res. C Emerg. Technol.*, 1996, **4**, (5), pp. 307–318

[5] Lee, S., Fambro, D.: 'Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting', *Transport. Res. Rec.*, 1999, **1678**, pp. 179–188

[6] Peter, Ä., Silvia, P.: 'ARIMA vs. ARIMAX-which approach is better to analyze and forecast macroeconomic time series'. Proc. 30th Int. Conf. Mathematical Methods in Economics. Karviná, Czech Republic, 2012

[7] Mukherjee, S., Osuna, E., Girosi, F.: 'Nonlinear prediction of chaotic time series using support vector machines'. Neural Networks for Signal Processing [1997] VII. Proc. 1997 IEEE Workshop. IEEE, Amelia Island, FL, USA, September 1997

[8] Zhang, Y., Liu, Y.: 'Traffic forecasting using least squares support vector machines', *Transportmetrica*, 2009, **5**, (3), pp. 193–213

[9] Bing, Q., Gong, B., Yang, Z., *et al.*: 'Short-term traffic flow local prediction based on combined kernel function relevance vector machine model', *Math. Probl. Eng.*, 2015, pp. 1–9

[10] Kotsiantis, S.B., Zaharakis, I., Pintelas, P.: 'Supervised machine learning: a review of classification techniques', 2007, pp. 3–24

[11] Loh, W.-Y.: 'Classification and regression trees', *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, 2011, **1**, (1), pp. 14–23

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

[13] Vlahogianni, E.I., Golias, J.C., Karlaftis, M.G.: 'Short-term traffic forecasting: overview of objectives and methods', *Transp. Rev.*, 2004, **24**, (5), pp. 533–557

[14] Mori, U, Mendiburu, A, Ālvarez, M, *et al.*: 'A review of travel time estimation and forecasting for advanced traveler information systems', *Transportmetrica A Transport Sci.*, 2015, **11**, (2), pp. 119–157

[15] Seabold, S., Perktold, J.: 'Statsmodels: econometric and statistical modeling with python'. 9th Python in Science Conf., Austin, TX, USA, June 2010

[16] Washington, S., Karlaftis, M.G., Mannering, F.L.: '*Statistical and econometric methods for transportation data analysis*' (Chapman & Hall/CRC Press, Boca Raton, 2010, 2nd edn.)

[17] Tsirigotis, L., Vlahogianni, E.I., Karlaftis, M.G.: 'Does information on weather affect the performance of short-term traffic forecasting models?' *Int. J. Intell. Transport. Syst. Res.*, 2012, **10**, (1), pp. 1–10

[18] Müller, K.R., Smola, A.J., Rätsch, G., *et al.*: 'Predicting time series with support vector machines', in Gerstner, W., Germond, A., Hasler, M., Nicoud, J. D. (eds.) *Artificial neural networks-ICANN'97. ICANN 1997. Lecture notes in computer science*, vol. **1327** (Springer, Berlin, Heidelberg, 1997), pp. 999–1004

[19] Basak, D., Pal, S., Patranabis, D.C.: 'Support vector regression', *Neural Inf. Process. Lett. Rev.*, 2007, **11**, (10), pp. 203–224

[20] das Chagas Moura, M., Zio, E., Lins, I.D., *et al.*: 'Failure and reliability prediction by support vector machines regression of time series data', *Reliabil. Eng. Syst. Saf.*, 2011, **96**, (11), pp. 1527–1534

[21] Hou, Y., Edara, P., Sun, C.: 'Traffic flow forecasting for urban work zones', *IEEE Trans. Intell. Transp. Syst.*, 2015, **16**, (4), pp. 1761–1770

[22] Breiman, L.: 'Random forests', *Mach. Learn.*, 2001, **45**, pp. 5–32

[23] Friedman, J., Hastie, T., Tibshirani, R.: '*The elements of statistical learning*' vol. **1** (Springer Series in Statistics, New York, 2001)

[24] Piramu Kailasam, S., Aruna Anushiya, K., Mohammed Sathik, M.: 'Traffic flow prediction with big data using SAE'S algorithm', *IJCSMC*, 2016, **5**, (7), pp. 186–193

[25] Sutskever, I., Martens, J., Hinton, G.E.: 'Generating text with recurrent neural networks'. Proc. Int. Conf. Machine Learning (ICML-11). Bellevue, WA, USA, June 2011

[26] Lv, Y., Duan, Y., Kang, W., *et al.*: 'Traffic flow prediction with big data: a deep learning approach', *IEEE Trans. Intell. Transp. Syst.*, 2015, **16**, (2), pp. 865–873

[27] Ma, X., Yu, H., Wang, Y., *et al.*: 'Large-scale transportation network congestion evolution prediction using deep learning theory', *PLoS One*, 2015, **10**, (3), p. e0119044

[28] Gers, F.: 'Long short-term memory in recurrent neural networks'. Unpublished PhD dissertation, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, 2001

[29] Chung, J., Gulcehre, C., Cho, K., *et al.*: 'Empirical evaluation of gated recurrent neural networks on sequence modeling', arXiv preprint arXiv:1412.3555, 2014

[30] Fu, R., Zhang, Z., Li, L.: 'Using LSTM and GRU neural network methods for traffic flow prediction'. Chinese Association of Automation (YAC), Youth Academic Annual Conf. IEEE, Wuhan, China, November, 2016

[31] Zhao, Z., Chen, W., Wu, X., *et al.*: 'LSTM network: a deep learning approach for short-term traffic forecast', *IET Intell. Transp. Syst.*, 2017, **11**, (2), pp. 68–75

[32] Jia, Y., Wu, J., Xu, M.: 'Traffic flow prediction with rainfall impact using a deep learning method', *J. Adv. Transp.*, 2017, **2017**, pp. 1–10

[33] Guo, J.: '*Backpropagation through time*' Unpublished manuscript (Harbin Institute of Technology, 2013)

[34] Abadi, M., Barham, P., Chen, J., *et al.*: 'Tensorflow: a system for large-scale machine learning', *OSDI*, 2016, **16**, pp. 265–283

[35] Available at https://keras.io/

[36] Kingma, D., Ba, J.: 'Adam: a method for stochastic optimization'. arXiv preprint arXiv:1412.6980, 2014

[37] 'Caltrans Performance Measurement System'. Available at http://pems.dot.ca.gov/, 23 December 2011

[38] 'National Oceanic and Atmospheric Administration'. Available at https://www.ncdc.noaa.gov/cdo-web/datasets

[39] Rupp, M., Tkatchenko, A., Müller, K., *et al.*: 'Fast and accurate modeling of molecular atomization energies with machine learning', *Phys. Rev. Lett.*, 2012, **108**, (5), p. 058301

[40] Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: 'Extreme learning machine: theory and applications', *Neurocomputing*, 2006, **70**, (1), pp. 489–501

[41] Pedregosa, F., Varoquaux, G., Gramfort, A., *et al.*: 'Scikit-learn: machine learning in python', *J. Mach. Learn. Res.*, 2011, **12**, pp. 2825–2830

[42] Polson, N.G., Sokolov, V.O.: 'Deep learning for short-term traffic flow prediction', *Transp. Res. C, Emerg. Technol.*, 2017, **79**, pp. 1–17