A novel residual graph convolution deep learning model for short-term network-based traffic forecasting

Abstract: Short-term traffic forecasting on large street networks is significant in transportation and urban management, such as real-time route guidance and congestion alleviation. Nevertheless, it is very challenging to obtain high prediction accuracy with reasonable computational cost due to the complex spatial dependency on the traffic network and the time-varying traffic patterns. To address these issues, this paper develops a residual graph convolution long short-term memory (RGC-LSTM) model for spatial-temporal data forecasting considering the network topology. This model integrates a new graph convolution operator for spatial modelling on networks and a residual LSTM structure for temporal modelling considering multiple periodicities. The proposed model has few parameters, low computational complexity, and a fast convergence rate. The framework is evaluated on both the 10-min traffic speed data from Shanghai, China and the 5-min Caltrans Performance Measurement System (PeMS) traffic flow data. Experiments show the advantages of the proposed approach over various state-of-the-art baselines, as well as consistent performance across different datasets.

Keywords: short-term traffic forecasting; spatial-temporal dependency; network topology; graph convolution; residual long short-term memory

1 Introduction

Short-term traffic forecasting, e.g. traffic speed, flow and occupancy, aims to leverage historical traffic information to predict traffic conditions ahead of time, usually less than 15 minutes (Vlahogianni et al., 2014). Reliable, accurate and real-time prediction of spatio-temporal traffic data is significant for intelligent transportation systems, urban management, public safety, and even economic development (Weisbrod et al., 2003; Vlahogianni et al., 2014). For example, commercial transportation service providers, such as Google Map and Uber, require traffic data to provide online route guidance to save travellers' time. In addition, advanced intelligent traffic systems rely on traffic data to develop robust and smart traffic control strategies to alleviate congestion, which potentially helps to increase the number of jobs available and the productivity of urban areas.

Existing short-term traffic prediction models can be categorised into two types: statistical models and machine learning algorithms. A representative family of statistical models is the autoregressive integrated moving average (ARIMA) (Shu et al., 2003) and its diverse variants, including space-time ARIMA (Ding et al., 2011), dynamic space-time ARIMA (Min et al., 2009), and localised space-time ARIMA (Cheng et al., 2014). Other statistical models extensively used in short-term traffic forecasting include the hidden Markov model (Yu et al., 2003), Bayesian network (Sun et al., 2006), Kalman filter model (Guo et al., 2014), and the vector autoregressive model (Chen et al., 2016). An alternative modelling approach uses machine learning algorithms, such as k-nearest neighbour (Cai et al., 2016), support vector regression (Castro-Neto et al., 2009; Haworth et al., 2014; Feng et al., 2018), and neural networks (Dia, 2001; Vlahogianni et al., 2007; Wei and Chen, 2012; Kumar et al., 2013). However, the statistical models usually suffer from high computational complexity due to parameter inference. The classical machine learning models are shallow in

architecture, which limits the forecasting accuracy and reliability, due to the stochastic and nonlinear nature of traffic variables, especially in citywide traffic networks (Cui et al., 2018).

Lately, deep learning (DL), a state-of-the-art machine learning technique, has drawn great attention to complex spatial-temporal data modelling (Shaw et al., 2016; Zhang and Cheng, 2019a). Classical DL techniques have been used for traffic forecasting, including recurrent neural network (RNN) (Van Lint et al., 2002), long-short term memory (LSTM) (Tian and Pan, 2015), and stacked autoencoder network (Lv et al., 2015). While these models learn the implicit temporal relations of the data, the spatial dependency cannot be modelled explicitly. To take full advantage of the spatial dependency, Zhang et al. (2016) divided the traffic network onto grids and mapped traffic condition into grids as an image. The convolutional neural network (CNN) was then fed with sequential images as input to extract spatio-temporal features. It was widely successful in citywide traffic forecasting because CNN contains parameter sharing scheme to model spatial dependency in a large area with a small number of trainable parameters. Inspired by this initial work, many hybrid frameworks have been proposed. For instance, the combination of CNN, which captures spatial dependency and RNN/LSTM, which captures temporal dependency have recently become favoured approaches (Yu et al., 2017b; Zhao et al., 2017; Ren et al., 2019a). Similar works can also be found in the literature (Ke et al., 2017; Chen et al., 2018; Cheng et al., 2018; Liao et al., 2018; Liu et al., 2018a; Yao et al., 2018).

However, all the above-mentioned DL methods are based on two-dimensional grids, overlooking the topology of the underlying traffic network, since the classical CNN only works on regular Euclidean spaces, but not network-structured data. It has been suggested that network-based traffic forecasting is more reasonable and useful for several reasons (Cheng et al., 2012; Li and Shahabi, 2018; Zhang and Cheng, 2019b). First, the grid-based representation may not capture the correct spatio-temporal dependency of traffic data. For

example, two road segments in different directions of a road, although close in Euclid space, can have significantly different traffic conditions due to the network topology. Previous work has demonstrated the spatio-temporal autocorrelation of traffic data on transportation networks (Cheng et al., 2012). In addition, many potential applications rely on network-based traffic forecasting, such as route planning. These reasons highlight why the network-based method may provide superior traffic forecasting.

Recently, researchers have attempted to use DL for network-based traffic prediction. Studies have utilised a spectral graph convolution proposed by Defferrard et al. (2016) to extract spatial features from graph-structured data, along with employing other DL structures, such as RNN, to forecast traffic variables (Yu et al., 2017a; Lin et al., 2018; Zhang et al., 2019). However, the spectral graph convolution required the Chebyshev approximation to efficiently model high-order spatial dependency, which probably decreased the accuracy of spatial modelling. To overcome this issue, Cui et al. (2018) proposed a high-order graph convolutional LSTM model, which preserved the topology information via a weighted adjacency matrix of the traffic network. However, the spatial dependency captured by the weighted adjacency matrix was manually specified using the distance between locations rather than being adaptively learned by the model. Furthermore, the LSTM structure was hard to train, especially on large road networks. A similar issue exists in the gated graph RNN proposed by Wang et al. (2018), and it overlooked the multiple periodicities of traffic data for accurate temporal dependency modelling. Very recently, Ren et al. (2019b) utilised the adjacency relationship between road segments to construct a locally-connected deep network for road traffic forecasting. This model was powerful at preserving the topology information, but the number of trainable parameters linearly increased with the number of road segments, which limited the scalability of the model on large-scale traffic networks.

In view of the above models, the main challenges of network-based traffic prediction using DL are summarised into three points. (1) For spatial modelling, it is hard to explicitly model the spatial dependency considering the complex network topology. (2) For temporal modelling, classical RNN-based structures, including LSTM, are difficult to train. Moreover, extremely long temporal dependency is difficult to model, such as daily/weekly periodicities, and weekday/weekend traffic patterns. (3) The scalability of DL models for spatio-temporal traffic forecasting on large-scale networks has become a big concern. A more efficient DL structure with a small number of trainable parameters should be explored for fast training convergence required for practical applications.

To overcome these issues, this paper aims to model the complex spatio-temporal dependency of network-structured data rather than grid-structured data. A traffic network is modelled as a graph and a residual graph convolution LSTM (RGC-LSTM) is developed to capture the network-based spatial dependency and time-varying traffic patterns for traffic forecasting. It uses ~~a~~ few parameters, has low computational cost, and a fast convergence rate.

The RGC-LSTM is developed by integrating a novel graph convolution operator with a residual LSTM structure. First, the graph convolution operates on the weighted adjacency matrix of the graph and employ a parameter sharing scheme to learn localised spatial dependency considering the topology of the traffic network. The trainable parameters in the graph convolution operator are independent of the traffic network size and the computational complexity is just proportional to the number of graph edges. Second, the residual LSTM is utilised to model temporal dependency considering multiple periodicities and weekday/weekend patterns, which extracts temporal features via a deep structure with a fast convergence rate. The adopting parameter sharing scheme of the graph convolution substantially reduces the number of training parameters and the deep structure of the residual

LSTM greatly accelerates the convergence. The integration addresses the issue of scalability for city-wide forecasting. All of the above are the theoretical contributions of this work. In addition, from an empirical perspective, we evaluate our model using a traffic speed dataset on urban arterials and a traffic flow dataset on freeways. Experiments show the proposed model outperforms various state-of-the-art traffic prediction baselines across different datasets.

The remainder of this paper is organised as follows. Section 2 elaborates on the proposed methods. Section 3 presents the case study using two large real datasets to validate the capability of the model. Section 4 provides an explicit discussion of the effectiveness and robustness of the proposed model. Finally, Section 5 summarises the conclusions and directions for further research.

2 Methodology

This section elaborates on the proposed RGC-LSTM model for short-term network-based traffic forecasting. The problem can be formulated as learning a mapping function $F(\cdot)$ which uses historical traffic observations as the input features (denoted as $\boldsymbol{X}_t$) to estimate the traffic conditions at the next time t given the topology $\mathcal{G}$ of the traffic network, written as:

$$Y_t = F(\boldsymbol{X}_t; \mathcal{G}) \tag{1}$$

In the following subsections, the traffic network is represented as a graph, which preserves the network topology information. Next, the proposed graph convolution operator and the residual LSTM are illustrated to model temporal and spatial dependencies, respectively. Finally, the RGC-LSTM is constructed by integrating the two components for spatial-temporal data forecasting.

2.1 Traffic Network as a Graph

This paper represents a traffic network as a graph $\mathcal{G} = (V, \varepsilon)$ with self-loops, where V is a set of prediction locations with $|V| = N$, and $\mathcal{E}$ is a set of edges including the self-loops. The self-loops indicate that the future traffic condition at a specific location is self-affected by its historical observations. The links between any two different nodes indicate the two locations are adjacent, implying that the traffic condition of a location is affected by the traffic condition of nearby locations.

2.2 Graph Convolution for Spatial Dependency Modelling

Spatial dependency modelling on graphs is challenging due to the complex graph topology. Borrowing the idea of parameter sharing from CNN, this article proposes a graph convolution operator, which acts on the weighted adjacency matrix of a graph for efficient spatial dependency modelling with low computational complexity and ~~a~~ few trainable parameters. Details are given below.

2.2.1 Weighted Adjacency Matrix

Spatial dependency modelling on a graph relies on the topology captured by the adjacency matrix. However, the spatial dependency of traffic conditions between different locations may vary across the entire graph. This paper proposes to use a weighted adjacency matrix $W \in \mathbb{R}^{N \times N}$ to capture the heterogeneous spatial correlations, so that parameter sharing can be further adopted for adaptive spatial modelling on graphs.

The weight of a graph edge is defined by calculating the temporal correlation coefficient using histroical traffic observations. To calculate the coefficient, we first use the Min-Max method to rescale the observations at each location into the same range [0, 1]. This is because the maximum traffic observations (e.g., flow or speed) on a road is restricted by its physical characteristics (e.g., capacity or speed limit). Suppose $\boldsymbol{x}_i =$

$\left(x_{11}^i, x_{21}^i, \cdots, x_{T1}^i, \cdots, x_{td}^i, \cdots, x_{TD}^i\right)$ represents a time series, where x_{td}^i is the traffic condition of the *i*-th location $(1 \leq i \leq N)$ measured during time interval *t* $(0 \leq t < T)$ on day *d* $(0 \leq d < D)$. The traffic data after min-max scaling is written as:

$$\tilde{x}_{td}^i = \frac{x_{td}^i - x_{min}^i}{x_{max}^i - x_{min}^i} \tag{2}$$

where x_{max}^i and x_{min}^i are the maximum and minimum observations of the *i*-th location, respectively.

Second, the daily periodicity is removed from the traffic time series using z-score transformation, because it may result in strong temporal autocorrelation which amplifies the temporal similarity between locations (Yang et al., 2017). The z-score normalisation is:

$$\hat{x}_{td}^i = \frac{\tilde{x}_{td}^i - \bar{x}_t^i}{\sigma_t^i} \tag{3}$$

where $\bar{x}_t^i = \frac{1}{D}\sum_{d=0}^{D-1} \tilde{x}_{td}^i$ and $\sigma_t^i = \sqrt{\frac{1}{D-1}\sum_{d=0}^{D-1}\left(\tilde{x}_{td}^i - \bar{x}_t^i\right)^2}$ are the mean and the standard deviation of the rescaled traffic of the *i*-th locations at time *t*, respectively.

The similarity of any two adjacent locations is then calculated as

$$r_{ij} = r\left(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_j\right) = \frac{\sum_{d=0}^{D-1}\sum_{t=0}^{T-1}\left(\hat{x}_{td}^i - \bar{x}_i'\right)\left(\hat{x}_{td}^j - \bar{x}_j'\right)}{\sum_{d=0}^{D-1}\sum_{t=0}^{T-1}\left(\hat{x}_{td}^i - \bar{x}_i'\right)^2 \sum_{d=0}^{D-1}\sum_{t=0}^{T-1}\left(\hat{x}_{td}^j - \bar{x}_j'\right)^2} \tag{4}$$

where $\bar{x}_i'$ and $\bar{x}_j'$ are the means of normalised time series $\hat{\boldsymbol{x}}_i = \left(\hat{x}_{11}^i, \cdots, \hat{x}_{TD}^i\right)$ and $\hat{\boldsymbol{x}}_j = \left(\hat{x}_{11}^j, \cdots, \hat{x}_{TD}^j\right)$, respectively. The weighted adjacency matrix of the graph is then defined as:

$$w_{ij} = \begin{cases} r\left(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_j\right) & \text{if } (i,j) \text{ is an edge} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The matrix takes full advantage of the temporal trending of the traffic variables. It helps to make the spatial dependency more localised than the plain adjacent matrix, i.e., a graph node has more similar traffic conditions with its neighbouring nodes connected by an edge with

larger weights. Hence, graph convolution can operate on it by adopting a parameter sharing scheme.

2.2.2 A Novel Graph Convolution Operator

To capture the localised spatial dependency on graphs, a *K*-order graph convolution operator $*_{\mathcal{G}}$ is defined as the multiplication of a input $X = (x_1, \cdots, x_N) \in \mathbb{R}^N$ (i.e., a scalar for every node) with a polynomial graph filter g_θ operating on the weighted adjacency matrix, formulated as:

$$\begin{aligned} Y &= W *_{\mathcal{G}} X = g_\theta(W)X \\ &= \theta_0 W^0 X + \theta_1 W^1 X + \cdots + \theta_{K-1} W^{K-1} X = \sum_{k=0}^{K-1} \theta_k W^k X \end{aligned} \tag{6}$$

where $Y = (y_1, \cdots, y_N)$ denotes the outputs at each node, W^k is the *k*-th power of the weight matrix W and W^0 is an identity matrix of the same dimension as W. *K* is the order of the graph filter g_θ, also referred to as kernel size and $\boldsymbol{\theta} = (\theta_0, \cdots, \theta_{K-1}) \in \mathbb{R}^K$ is a vector of trainable parameters.

The proposed graph convolution operator can be easily interpreted from the vertex domain. According to the property of the weight matrix W, $W^K(i,j) = w_{ij}^{(K)} = 0$ when the distance of the shortest path between nodes *i* and *j* (i.e., the minimum number of edges comprising any path connecting *i* and *j*) is greater than *K*. Thus, the output Y at the *i*-th vertex after the graph convolution operation (i.e., the *i*-th row of Y in Eq. (6)) is:

$$
\begin{aligned}
y_i &= \theta_0 w_{ii}^{(0)} x_i + \theta_1 \sum_{j\in\mathcal{N}(i,1)} w_{ij}^{(1)} x_j + \cdots + \theta_{K-1} \sum_{j\in\mathcal{N}(i,K-1)} w_{ij}^{(K-1)} x_j \\
&= x_i \sum_{k=0}^{K-1} \theta_k w_{ii}^{(k)} + \sum_{j\in\mathcal{N}_i^1} x_j \left(\sum_{k=1}^{K-1} \theta_k w_{ij}^{(k)} \right) + \cdots + \sum_{j\in\mathcal{N}_i^{K-1}} x_j \theta_{K-1} w_{ij}^{(K-1)} \\
&= x_i \bar{\theta}_{ii}^{(0)} + \sum_{j\in\mathcal{N}_i^1} x_j \bar{\theta}_{ij}^{(1)} + \cdots + \sum_{j\in\mathcal{N}_i^{K-1}} x_j \bar{\theta}_{ij}^{(K-1)} = \sum_{n=0}^{K-1} \sum_{j\in\mathcal{N}_i^n} x_j \bar{\theta}_{ij}^{(n)}
\end{aligned}
\tag{7}
$$

where $w_{ij}^{(k)}$ is the entry of W^k at location $[i,j]$. $\mathcal{N}_i^k$, termed the k-order neighbourhood of node i, is the set of nodes connected to node i by the shortest path of k edges. $\mathcal{N}(i,k)$ is referred to as a k-hop local neighbourhood of node i, which denotes the set of nodes connected to node i by the shortest path of k or fewer edges. The above notations are further illustrated in Fig 1. In addition, $\bar{\theta}_{ij}^{(n)} = \sum_{k=n}^{K-1} \theta_k \, w_{ij}^{(k)}$ ($0 \le n \le K-1$ and $j \in \mathcal{N}_i^n$), is the coefficient used to multiply the input values at the n-order adjacent neighbour node j of node i. Thus, the output at node i is a linear combination of the inputs at nodes within the (K-1)-hop local neighbourhood of node i.

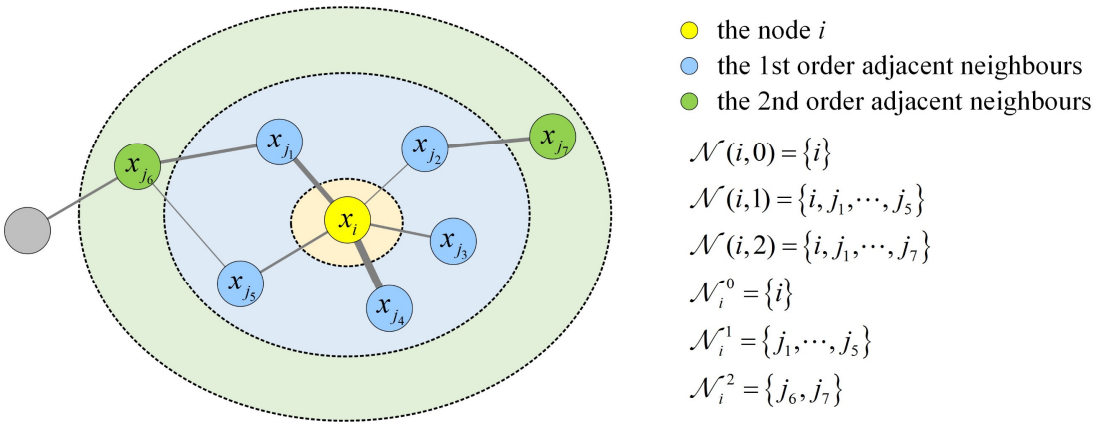

Fig 1. A graphical illustration of the notations in Eq. (7). For instance, $\mathcal{N}(i,1)$ contains all yellow and blues nodes (i.e., zero- and first-order neighbours).

The proposed graph convolution operation with different Ks can be visualised using a network structure, as shown in Fig 2. A K-order graph convolution equivalently captures the

spatial correlations between any node *i* and its *k*-order ($0 \leq k \leq K-1$) adjacent neighbours. In addition, although all nodes share the trainable parameters $\theta_0, \cdots, \theta_{K-1}$ in the graph convolution, the coefficient $\bar{\theta}_{ij}^{(n)}$ in the linear combination of any node i is different, since the weighted adjacency matrix captures the heterogeneity of the spatial dependency. Due to parameter sharing, the number of learnable parameters in the proposed graph convolution operation is notably reduced to $\mathcal{O}(K)$, which is independent of the number of edges or nodes in the graph.

However, the computational complexity of $W^k X$ is up to $\mathcal{O}(N^2)$, because W^k becomes dense as *k* increases. To reduce the computational cost, we propose to calculate the higher-order weight matrix recursively. Denoting $\bar{X}_k = W^k X \in \mathbb{R}^N$, we first calculate the $W^1 X$ using sparse matrix multiplication. As W ~~a~~ is always a sparse matrix, the computational complexity of $W^1 X$ is only $\mathcal{O}(|\varepsilon|)$, which is the number of non-zero elements in W. To calculate the $W^2 X$, instead of directly calculating W^2, we use $\bar{X}_1$ to compute the $W^2 X$, namely $W^2 X = W\bar{X}_1$. As W is sparse and $\bar{X}_1$ is a *N*-dimensional vector, the computational complexity of $W\bar{X}_1$ is still $\mathcal{O}(|\varepsilon|)$. In this way, we can recursively calculate the $W^k X = \bar{X}_k = W\bar{X}_{k-1}$ and the computational cost won't increase with the order *k* increasing, as $W\bar{X}_{k-1}$ can be efficiently implemented as a product of a sparse matrix with a dense vector. . Thus, the total computational complexity of a *K*-order graph convolution operation is $\mathcal{O}(K|\varepsilon|) \ll \mathcal{O}(N^2)$, i.e., linear in the number of edges in the graph.

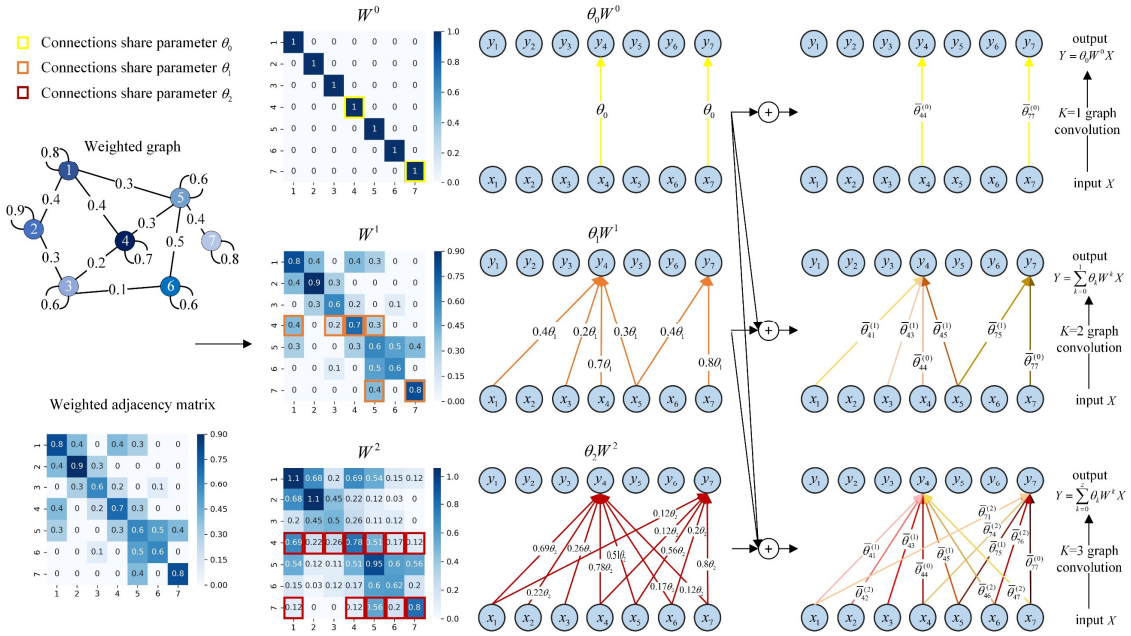

Fig 2. Graph convolution operation using graph filters with kernel size K can access neighbour nodes at most at K-1 hops. Nodes 4 and 7 are taken as examples to show that the parameter sharing scheme (in the third-column plots, arrow lines with the same colour share the learnable parameter) for spatial dependency modelling while the weighted adjacency captures the spatial heterogeneity.

2.3 Residual LSTM for Temporal Dependency Modelling

This paper utilises an LSTM-based structure to model temporal dependency. Traditional LSTM-based temporal dependency modelling has two issues. First, traffic data exhibit multiple periodicities and different patterns on weekday/weekends, which is difficult for LSTM structure to model. Second, training of LSTM becomes difficult as the depth of the DL model increases. To tackle these issues, we propose a new method of temporal feature construction to capture multiple traffic periodicities and patterns and we also introduce a residual LSTM structure to speed up the convergence of model training.

2.3.1 Temporal Feature Construction

Studies have shown that traffic has near-term, daily and weekly periodicities (Wu and Tan, 2016). In addition, weekend traffic patterns are quite distinct from those on weekdays. This

article proposes to integrate near-term, daily, and weekly temporal features as the input for temporal dependency modelling. To distinguish weekday and weekend traffic patterns, we also propose a new way to take account into the day of the week when constructing the daily feature.

Concretely, at a prediction time *t*, the near-term features refer to the historical traffic data observed during time $[t-S, t-1]$, denoted as $[X_{t-S}, X_{t-S+1}, \cdots, X_{t-1}]$, where $X_i = [x_i^1, x_i^2, \cdots, x_i^N]^T \in \mathbb{R}^{N\times 1}$ are the observations of the *N* prediction locations at time interval *i* and x_i^j is the traffic condition at the *j*-th location. The length of the near-term feature *S* is termed the time-window.

Next, the daily feature of a specific prediction time on a weekday/weekend, denoted as $X_{t,d}$, is the historical traffic condition observed at the same time on the last weekday/weekend. For example, as shown in Fig 3, to predict the traffic at 12:00 pm on Monday, we use the historical observation at the same time interval on the last Friday, not Sunday, as the daily features. To predict Saturday traffic flow at 12: 00 pm, the historical traffic data at the same time on the last Sunday serves as the daily feature. For any other day of week, the historical observation at the same time on the previous day is used as the daily feature. This method is visualised in Fig 3 using the curve arrows below the X-axis.

In addition, the weekly feature $X_{t,w}$ is the historical traffic flow at the same time on the same day of the last week. In this way, the temporal features as input for temporal dependency modelling is rewritten as:

$$\boldsymbol{X}_t = \left[X_{t,w}, X_{t,d}, X_{t-S}, \cdots, X_{t-1}\right] = \begin{bmatrix} x_{t,w}^1 & x_{t,d}^1 & x_{t-S}^1 & \cdots & x_{t-1}^1 \\ x_{t,w}^2 & x_{t,d}^2 & x_{t-S}^2 & \cdots & x_{t-1}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{t,w}^N & x_{t,d}^N & x_{t-S}^N & \cdots & x_{t-1}^N \end{bmatrix} \tag{8}$$

where $x_{t,d}^j$ and $x_{t,w}^j$ denote the daily and weekly periodic features, respectively. Each X_i is referred to as a graph signal.

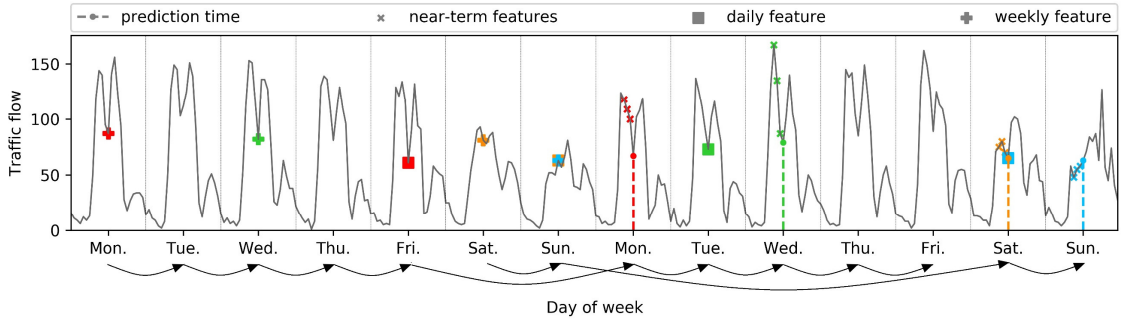

Fig 3. Examples of input feature construction. Historical observations labelled by markers with the same colour are used as the features to predict the traffic flow at the chosen prediction time.

2.3.2 Residual LSTM

Classical LSTM is capable and robust for sequential data modelling. With an input $\boldsymbol{x} = (x_0, x_1, \cdots, x_{T_s-1})$, a classical LSTM generates a sequence of output denoted as $\boldsymbol{h} = (h_0, h_1, \cdots, h_{T_s-1})$ as an estimation of the actual value $\boldsymbol{y} = (y_0, y_1, \cdots, y_{T_s-1})$, where T_s is referred to as time-step. LSTM can be unrolled like a chain consisting of T_s LSTM units, as shown in Fig 4. One of the most popular classical LSTM units (Graves, 2013) is formulated as:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + w_{ci} \odot c_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + w_{cf} \odot c_{t-1} + b_f) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + w_{co} \odot c_t + b_o) \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{9}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, $\odot$ is the pointwise product, and $\tanh(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$ is the activation function. The vector variables $x_t \in \mathbb{R}^{d_x}$, $h_t \in [-1,1]^{d_h}$ and $c_t \in \mathbb{R}^{d_h}$ represent the unit input, output, and internal states at time step *t*, respectively. d_h, referred to as the number of hidden units, refers to the dimension of the 'hidden state' of the

LSTM layer. In addition, i_t, f_t , and $o_t \in [0,1]^{d_h}$ are the input, forget, and output gate, which determine whether passing a new input, blocking the current state, and letting the current state affect the output at each time step, respectively. The weights $W_{x\bullet} \in \mathbb{R}^{d_h \times d_x}$, $W_{h\bullet} \in \mathbb{R}^{d_h \times d_h}$, $w_{c\bullet} \in \mathbb{R}^{d_h}$ and the four biases $b_\bullet \in \mathbb{R}^{d_h}$ are the intercept parameters of the unit.

However, the classical structure of LSTM leads to a difficult and slow training process. A residual LSTM is used to speed up convergence in order to decrease training time. The residual learning was first developed for CNN models (He et al., 2016) to address gradient degradation issues and accelerate the convergence of the model training process. Assuming the input and the output are of the same dimension, instead of learning a direct mapping $F(\cdot)$ from the input x to the output y, i.e., $y = F(x)$, a residual model learns a residual function $F_{res}(\cdot)$ satisfying

$$y = F_{res}(x) + x \tag{10}$$

where $F_{res}(\cdot)$ fits the residual part $y - x$ (the difference between the observed value and the actual value), and the second term x in Eq. (10) is termed an identity mapping. This paper implements the residual learning on LSTM by adding an identity mapping via a shortcut from the input to the output in each LSTM unit. The inner structure of a residual LSTM unit is visualised in Fig 4 (B) and the shortcut for identity mapping is highlighted by the red arrow. A residual LSTM unit has the same first four formulas of i_t, f_t, c_t, and o_t, as defined in Eq. (9) but the last formula for h_t is rewritten as:

$$h_t = o_t \odot \tanh(c_t) + W_{trans} x_t \tag{11}$$

where $W_{trans} \in \mathbb{R}^{d_h \times d_x}$ is used to match the dimensions of h_t and x_t. Essentially, a residual LSTM unit is a classical LSTM unit with identity mapping. The output of a classical LSTM unit formulated as $o_t \odot \tanh(c_t)$ serves as the residual function $F_{res}(\cdot)$, and $W_{trans} x_t$ serves as an identity mapping, which directly passes the linear projection of the input x_t to the

output via the shortcut. This change implies that a residual LSTM unit learns a residual function with reference to the unit input.

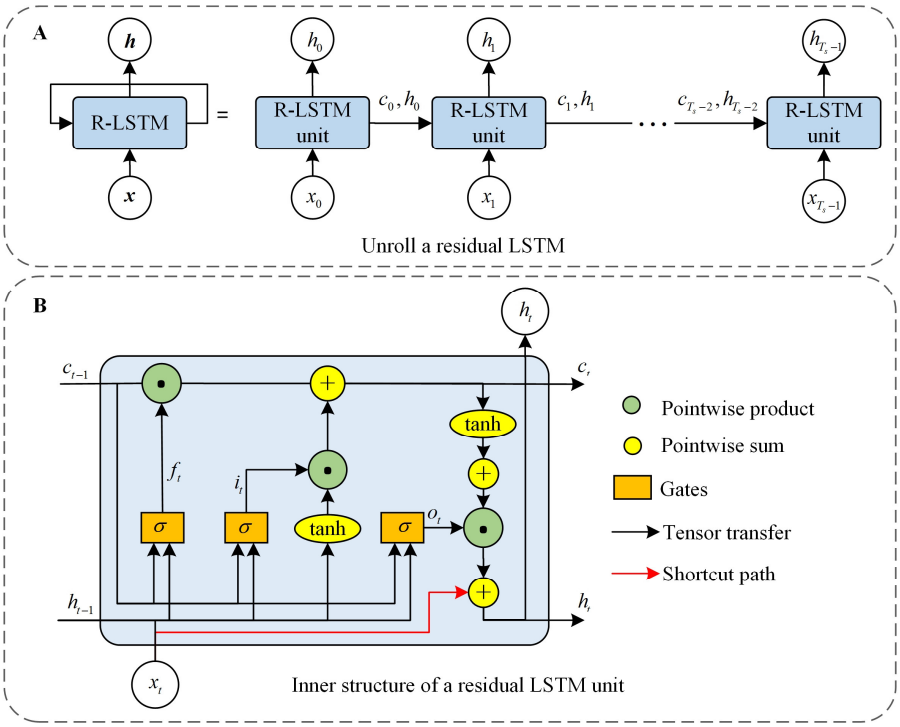

Fig 4. (A) A chain-like unrolled residual LSTM. (B) The inner structure of a residual LSTM unit, which has one more shortcut path (highlight in red) for identity mapping than classical LSTM unit.

2.4 RGC-LSTM Network

To model the spatial-temporal dependencies of network-based data, a spatial-temporal deep learning framework is developed by stacking multiple RGC-LSTM layers and a fully-connected layer to form a deep structure, as shown in Fig 5. Each RGC-LSTM layer integrates the residual LSTM and the proposed graph convolution operations.

In each RGC-LSTM layer, the proposed graph convolution is used to make the residual LSTM suitable for graph-structured data in order to automatically extract the spatial-temporal features with a fast convergence rate. To do so, the multiplications in the residual LSTM are replaced with the graph convolution operation $* \mathcal{G}$ defined in Eq. (6). Suppose that the input of the l-th RGC-LSTM layer is a sequence of graph signals denoted as $\boldsymbol{X}^l =$

$\left(X_0^l, X_1^l, \cdots X_{T_s-1}^l\right)$ with $X_t^l \in \mathbb{R}^N$, where T_s is the number of time steps and N is the number of graph nodes. In this paper, T_s equals the total length of the input near-term, daily, and weekly temporal sequence, i.e., $T_s = S + 2$. An RGC-LSTM layer generates a sequence output denoted as $\boldsymbol{h}^l = \left(h_0^l, h_1^l, \cdots h_{T_s-1}^l\right)$. The formulation of an RGC-LSTM unit at the time-step t $(0 \le t \le T_s - 1)$ in the l-th layer is:

$$
\begin{aligned}
i_t^l &= \sigma\left(W_{xi}^l *_{\mathcal{G}} X_t^l + W_{hi}^l *_{\mathcal{G}} h_{t-1}^l + w_{ci}^l \odot c_{t-1}^l + b_i^l\right) \\
f_t^l &= \sigma\left(W_{xf}^l *_{\mathcal{G}} X_t^l + W_{hf}^l *_{\mathcal{G}} h_{t-1}^l + w_{cf}^l \odot c_{t-1}^l + b_f^l\right) \\
c_t^l &= f_t^l \odot c_{t-1}^l + i_t^l \odot tanh\left(W_{xc}^l *_{\mathcal{G}} X_t^l + W_{hc}^l *_{\mathcal{G}} h_{t-1}^l + b_c^l\right) \\
o_t^l &= \sigma\left(W_{xo}^l *_{\mathcal{G}} X_t^l + W_{ho}^l *_{\mathcal{G}} h_{t-1}^l + w_{co}^l \odot c_t^l + b_o^l\right) \\
h_t^l &= o_t^l \odot tan\, h\left(c_t^l\right) + W_{trans}^l *_{\mathcal{G}} x_t
\end{aligned}
\tag{12}
$$

where K^l is the order of the graph convolution, $W_{x\bullet}^l \in \mathbb{R}^{K^l \times d_h^l}$ and $W_{h\bullet}^l \in \mathbb{R}^{K^l \times d_h^l}$ are learnable parameters, d_h^l is the number of graph filters in the l-th layer (also referred to as the number of hidden units), and $h_t \in \mathbb{R}^{N \times d_h^l}$ is the output at time-step t.

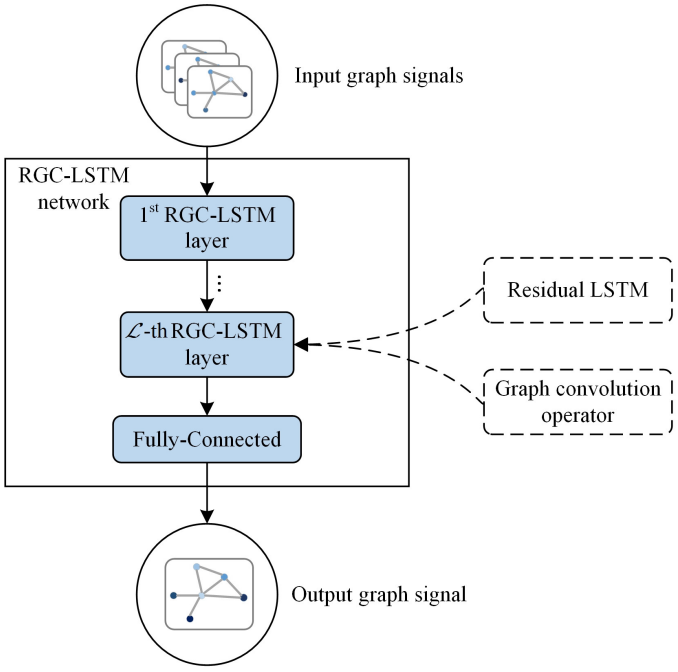

Fig 5. The structure of the proposed RGC-LSTM network.

There are two advantages of the integration. First, it can extract the spatial and temporal features simultaneously, not sequentially. Second, since the proposed graph convolution is a liner operator, this integration can introduce nonlinearity into the model in both spatial and temporal domains due to the nonlinear activation function σ. Thus, the model can capture the complex spatial-temporal dependency modelling on graphs.

The proposed RGC-LSTM network is then a stack of $\mathcal{L}$ RGC-LSTM layers and a fully-connected layer. Supposing the prediction time point is t_p, the formulation of the RGC-LSTM architecture for spatial-temporal short-term traffic prediction is:

$$\left(h_{t_p,w}^{\mathcal{L}}, h_{t_p,d}^{\mathcal{L}}, h_{t_p-S}^{\mathcal{L}}, \cdots, h_{t_p-1}^{\mathcal{L}}\right) = \mathcal{F}^{\mathcal{L}} \cdots \mathcal{F}^2 \mathcal{F}^1 \left(X_{t_p,w}, X_{t_p,d}, X_{t_p-S}, \cdots, X_{t_p-1}\right) \quad (13)$$

$$\hat{X}_{t_p} = W_f \cdot \left(h_{t_p-1}^{\mathcal{L}}\right)^T + b_f \quad (14)$$

In Eq. (13), $\mathcal{F}^i$ $(1 \leq i \leq \mathcal{L})$ denotes the *i*-th RGC-LSTM layer, and $\left(h_{t_p,w}^{\mathcal{L}}, h_{t_p,d}^{\mathcal{L}}, h_{t_p-S}^{\mathcal{L}}, \cdots, h_{t_p-1}^{\mathcal{L}}\right) \in \mathbb{R}^{T_s \times N \times d_h^{\mathcal{L}}}$ is the output of the $\mathcal{L}$-th RGC-LSTM layer. Eq. (14) represents the fully-connected layer with trainable parameters $W_f \in \mathbb{R}^{d_h^{\mathcal{L}}}$ and $b_f \in \mathbb{R}$, to yield the estimated network traffic condition $\hat{X}_{t_p} \in \mathbb{R}^N$.

3 Case Study

3.1 Experiment Settings

3.1.1 Dataset and Pre-Processing

Two open traffic datasets are used in this case study to validate the proposed model, as follows:

- Shanghai traffic speed (SHSpeed) data (Wang et al., 2018): this dataset contains a 10-min traffic speed data from 156 urban road segments in the central area of Shanghai (Fig 6 (a)), China, collected from April 1, 2015 to April 30, 2015.

- Caltrans Performance Measurement System (PeMS) data: This dataset contains 5-min traffic flow provided by PeMS, collected from January 1, 2015 to May 31, 2015 by 1681 loop detectors at the mainline of the freeways in Los Angeles (District 7), California, USA, as shown in Fig 6 (b).

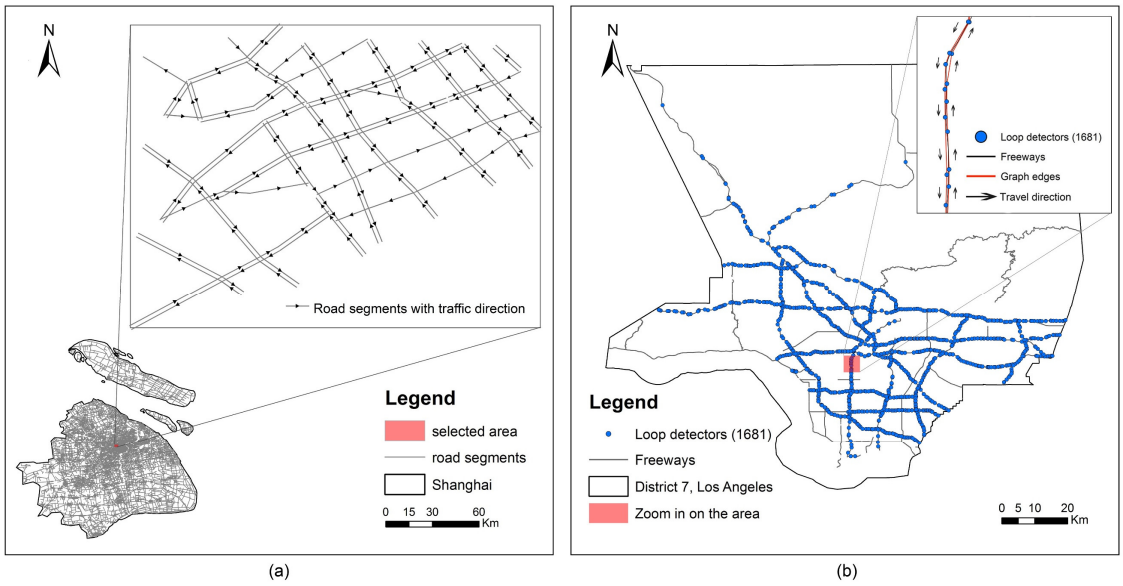

Fig 6. Two study areas. (a) An urban road network with 156 segments (zoom in upper right) in the central area of Shanghai, China. (b) The distribution of the 1681 loop detectors on the mainline of the freeways in District 7, California, USA. An area is selected to zoom in on to show the graph constructed by considering the adjacency relationship between detectors.

To construct the graph, each urban road segment in the first case and each traffic detector in the second case are treated as a graph node, separately. The graph topology is determined by the adjacency relationship between segments/detectors. Next, the two raw traffic datasets are scaled by segment/detector using the Min-Max method into the range [0, 1]. The input is a sequence of S near-term features, one daily, and one weekly feature, as defined in Eq. (8)). After constructing the input sequences, each dataset is divided into training, validation and testing parts, as detailed as in Table 1. The validation data are not used to train the model but to measure the model's predictive performance during the training process. Thus, we can only save the model's variables only if the performance is improved to

avoid overfitting the network to the training data, which weakens its generalisation to unseen data.

Table 1. Data division for model training, validation and testing.

Dataset	Total days	Training data (day)	Validation data (day)	Testing data (day)
SHSpeed	23	16	2	5
PeMS	144	100	14	30

3.1.2 Evaluation Metrics

The real traffic data (ground truth) are denoted as $(x_1, \cdots x_n)$ and the predicted values are denoted as $(x_1^{'}, \cdots x_n^{'})$. Three metrics are used to evaluate the prediction performance, namely Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). The definitions are as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - x_i^{'}\right)^2} \tag{16}$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|x_i - x_i^{'}\right| \tag{17}$$

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_i - x_i^{'}}{x_i}\right| \times 100\% \tag{18}$$

RMSE and MAE are used to evaluate the overall absolute prediction accuracy while MAPE is employed to measure the relative error.

3.2 Determine the Optimal RGC-LSTM Configuration

Three hyperparameters of the proposed RGC-LSTM network should be specified, including the number of RGC-LSTM layers $\mathcal{L}$, the number of hidden units in each RGC-LSTM layer, and the time-window S. Here, the number of hidden units corresponds to the number of graph filters. The grid search method is employed to determine the best parameter settings by changing one of the parameters while keeping the others unchanged. In this experiment, we

choose the number of layers $\mathcal{L}$ from 1 to 4, the number of hidden units from [4, 8, 16, 32], the order K from 1 to 5, and the time-window S from 1 to 5. To simplify this process, we use the same number of hidden units in each layer, and only change the graph convolution operator order K of the first layer while the others remain to be one.

During the training process, the mean square error is utilised as the loss function and variables are optimised by the Adam optimiser (Kingma and Ba, 2014). The learning rate is exponentially decayed from 0.15 at a rate of 0.96 per 50 steps. To ensure convergence, the number of training epochs and batch size are given in Table 2. After performing the grid search, the optimal configurations used in the two prediction tasks are presented in Table 2. More concrete discussions on the sensitivity analysis of the hyperparameters are provided in section 4.2.

Table 2. The training setting and the optimal configuration of the RGC-LSTM in the two forecasting tasks.

Dataset	Training epochs	Batch size	Layers	Hidden units	Time window S	Order K
SHSpeed	50	24	2	[16, 16]	4	2
PeMS	25	72	2	[8, 8]	3	3

3.3 Model Comparison

3.3.1 Baselines

The proposed RGC-LSTM is compared with six baseline models. Brief descriptions of the six benchmarks are given below.

- **ARIMA**: Autoregressive integrated moving average model (Box and Pierce, 1970), which is widely used for time series prediction.
- **SVR**: Support Vector Regression. The penalty parameter C and epsilon of the SVR are set to be 1 and 0.01, respectively.

- **LSTM**: A single layer LSTM models the temporal dependencies but not spatial relationships between locations.
- **CNN**: A CNN-based model proposed by Ma et al. (2017), which transforms the network traffic data into a two-dimensional image whose horizontal axis represents the prediction time and the vertical axis represents locations. Predictions are made by performing convolutions on the image.
- **STGCN**: A spatial-temporal graph convolutional network proposed by Yu et al. (2017a). Temporal features are extracted via gated convolution layers and spatial features are captured by graph convolution using the Laplacian matrix and its Chebyshev approximation for higher-order spatial dependency modelling.
- **GC-LSTM**: A multi-layer graph convolution LSTM model without residual learning short path. The parameter settings are the same as the proposed model except for the residual learning structure.

In the paper, the experiment environment is a workstation with 40 CPU cores (Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20 GHz (2 processors)), 256 GB RAM and one GPU (NVIDIA Quadro K2200). The ARIMA, SVR, and LSTM are implemented using Python 3.5.3 with package 'statsmodels', 'scikit-learn', and Keras with the TensorFlow backend, respectively. The RGC-LSTM model and the variant GC-LSTM are programmed using the same Python version and TensorFlow 1.2 (Abadi et al., 2016) with GPU support.

3.3.2 Comparison Results

Table 3 presents the performance comparison of the different algorithms. Overall, the proposed model outperforms the six baselines in terms of the three metrics in both prediction tasks. Compared to the optimal baseline, the RMSE, MAE, and MAPE of the proposed model are lower by 22.84%, 15.13% and 18.92%, respectively, on the SHSpeed data, and are

reduced by 10.97%, 12.67% and 33.99% respectively, on the PeMS data. In both cases, the MAPE is around 9%, which illustrates the consistent performance of the proposed model.

Table 3. Evaluation of the different models for the two traffic forecasting tasks

Dataset	Metrics	ARIMA	SVR	LSTM	CNN	STGCN	GC-LSTM	Ours
SHSpeed	RMSE	4.55	4.36	3.37	3.81	3.12	2.89	2.23
	MAE	3.27	3.09	2.42	2.73	2.23	2.18	1.85
	MAPE (%)	16.9	15.67	12.68	13.37	11.74	11.36	9.21
PeMS	RMSE	73.35	53.18	39.51	41.94	38.55	35.29	31.42
	MAE	57.63	37.13	29.49	31.88	28.42	24.71	21.58
	MAPE (%)	52.35	16.16	20.00	23.62	18.20	13.65	9.01

Concretely, it shows the simple statistic model ARIMA and the traditional machine learning model SVR perform worst. Among the deep learning models, LSTM only concentrates on temporal dynamic modelling while CNN mainly focuses on extracting features from the spatial domain. Overall, the former performs slightly better than the latter, which likely indicates that the temporal dependency modelling plays a more important role than the spatial dependency modelling in traffic forecasting. Another possible reason is that the CNN-based model proposed by Ma et al. (2017) is incapable of modelling the precise spatial relations regarding the complex network topology. However, when modelling the temporal and spatial dependency on graphs simultaneously, the last three graph-based spatial-temporal models perform better than LSTM and CNN.

In addition, STGCN is inferior to the last two LSTM-based models, probably because the LSTM is more powerful than the single gated structure in modelling the temporal dependency and dynamics of sequential data. In addition, STGCN utilises the Chebyshev approximation to reduce the computational complexity when modelling higher-order adjacency, which could decrease the precision of the spatial dependency modelling. The proposed model also outperforms its variant GC-LSTM, which validates the significance of the residual learning structure. To further highlight the advantage of the residual structure, the

validation loss and the accumulated training time of the GC-LSTM and the proposed model on PeMS data are illustrated in Fig 7. It shows the convergence of RGC-LSTM is faster than GC-LSTM and RGC-LSTM achieves a lower loss. In terms of training time, our model is slightly slower than GC-LSTM due to the identity mapping.

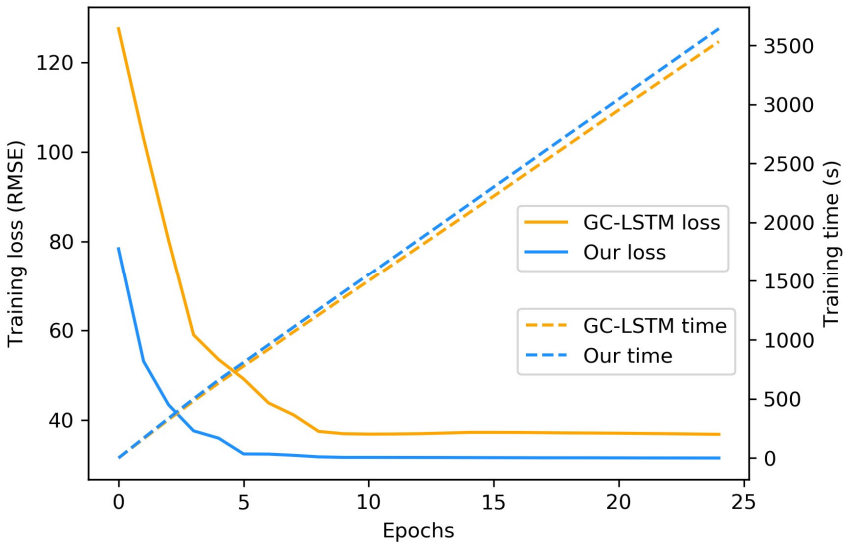

Fig 7. Training process comparison of GC-LSTM and RGC-LSTM network on PeMS data. The training loss is the RMSE. The accumulated runtime was drawn by the dashed line.

Fig 8 shows the boxplots of the RMSEs, MAEs and MAPEs produced by different models in the two cases. The prediction errors are calculated by road segment/traffic detector. It can be seen that the proposed model has the lowest outlier value in comparison with other baselines. In addition, it also demonstrates the advantage of the proposed model in terms of the maximum, minimum, and the median of prediction errors. Thus, the proposed model produces lower errors and has more stable performance than the baselines.

To further validate the significance of the accuracy improvement, a paired t-test is employed, referring to (Liu et al., 2018b). The null hypothesis for the two-sided paired t-test is that there is no difference between the prediction errors of the proposed RGC-LSTM and the benchmark, whereas the alternative hypothesis is that there is a significant difference. If the p-value is smaller than 0.05, the null hypothesis could be rejected. For brevity, this paper

only presents the results of the t-test on RMSEs by road segments/detectors in Table 4. The negative t-statistics demonstrate that the prediction errors of the proposed model are significantly smaller than that of the benchmarks. The significant difference also holds in terms of MAE and MAPE in both tasks.

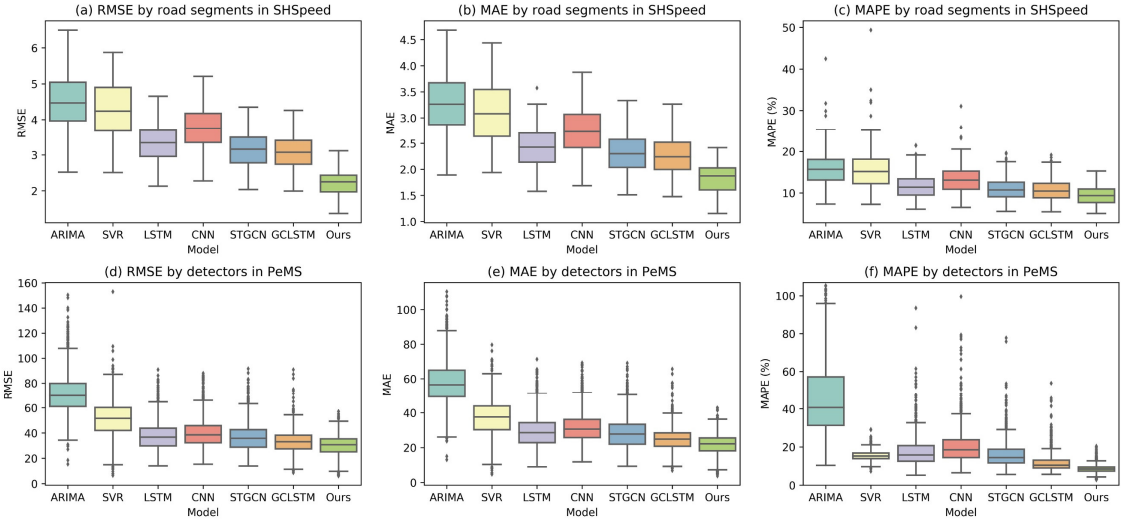

Fig 8 Comparison of forecast errors for each model on the PeMS data

Table 4. Two-side paired t-test results of the two traffic forecasting tasks

Dataset	Comparison	t-statistics	p-value	If significant
SHSpeed	Ours vs. ARIMA	-40.00	1.38e-83	Yes
	Ours vs. SVR	-33.01	5.16e-72	Yes
	Ours vs. LSTM	-15.38	5.23e-33	Yes
	Ours vs. CNN	-29.78	4.99e-66	Yes
	Ours vs. STGCN	-8.48	1.57e-14	Yes
	Ours vs. GC-LSTM	-5.34	3.24e-7	Yes
PeMS	Ours vs. ARIMA	-99.74	0.0	Yes
	Ours vs. SVR	-108.45	0.0	Yes
	Ours vs. LSTM	-48.22	2.79e-319	Yes
	Ours vs. CNN	-53.32	0.0	Yes
	Ours vs. STGCN	-79.00	0.0	Yes
	Ours vs. GC-LSTM	-44.91	4.58e-290	Yes

3.3.3 Graphical analysis

For a more concrete and direct comparison of the prediction performance, a road segment from the SHSpeed data and two typical detectors with low and high traffic flow in the PeMS dataset are selected to present the comparison of the forecasting results and the prediction errors, as shown in Fig 9. For clarity, only one-week of PeMS data from Monday to Sunday are visualised as an example. In each subfigure, the upper plot is the predicted traffic speed/flow of the seven different algorithms and the ground truth, while the lower plot presents the prediction errors deviated from the ground truth.

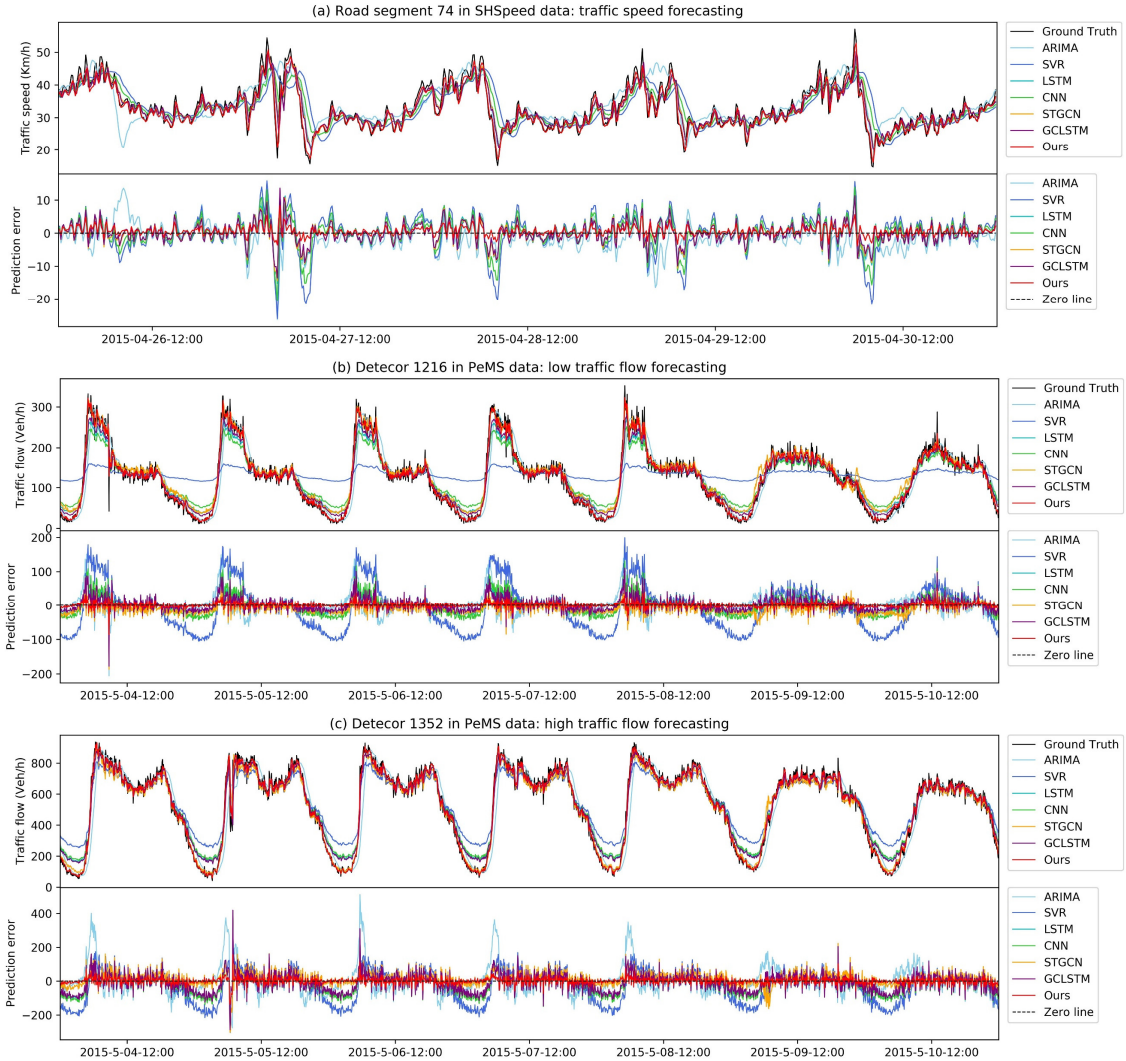

Fig 9. Comparison of traffic forecasting and prediction errors of road/detectors. (a) Traffic speed prediction of road segment 74 (b) Detector 1216 with low traffic flow. (c) Detector 1352 with high traffic flow.

It is shown the performance of the ARIMA and SVR vary greatly under different traffic conditions, because the parameters of ARIMA or SVR could not be manually tuned for each road/detector. This notably decreases the performance of ARIMAs and SVRs. Alternatively, the predicted traffic flow of deep learning models matches all traffic conditions quite well. Among all algorithms, RGC-LSTM can track the ground truth more promptly and its prediction is the closet to the ground truth. RGC-LSTM also predicts the peaks more correctly than the other baselines. In addition, by incorporating daily and weekly periodic features, the proposed model can accommodate both weekday and weekend traffic patterns. Hence, the proposed framework is effective and promising for traffic forecasting in practice.

4 Discussion

4.1 Prediction Error Analysis

In this section, we explore the distributions of the prediction errors of the RGC-LSTM in space and time. This analysis helps to further evaluate the performance of the proposed model.

4.1.1 Prediction Error Distributions in Space

The MAPE is selected as a representative metric for the assessment of the spatial distribution of prediction errors. Fig 10 displays the spatial distributions of MAPE by road segments in the SHSpeed data and by detectors in the PeMS data. In Fig 10 (a), all MAPEs are lower than 25% and 96.15% of road segments had MAPEs less than 15%. There are two segments of MAPE higher than 20%, located at the border of the study area. This could be

because the spatially adjacent neighbours are not entirely included in the study area, which implies the importance of spatial dependency modelling.

According to Fig 10 (b), all MAPEs of detectors are lower than 30% and those with MAPE values less than 15% account for 95.24% of all detectors. In addition, larger MAPE values often occur at detectors with low traffic flow, which are mainly deployed on the freeways away from the central area. Contrarily, most of the detectors deployed on very crowded freeways have MAPEs of less than 10%. In practice, considering that the predictions may be used to disperse traffic, alleviate congestion, or detect crowd flows to avoid catastrophic stampedes for public safety, the proposed method is promising in practical applications.

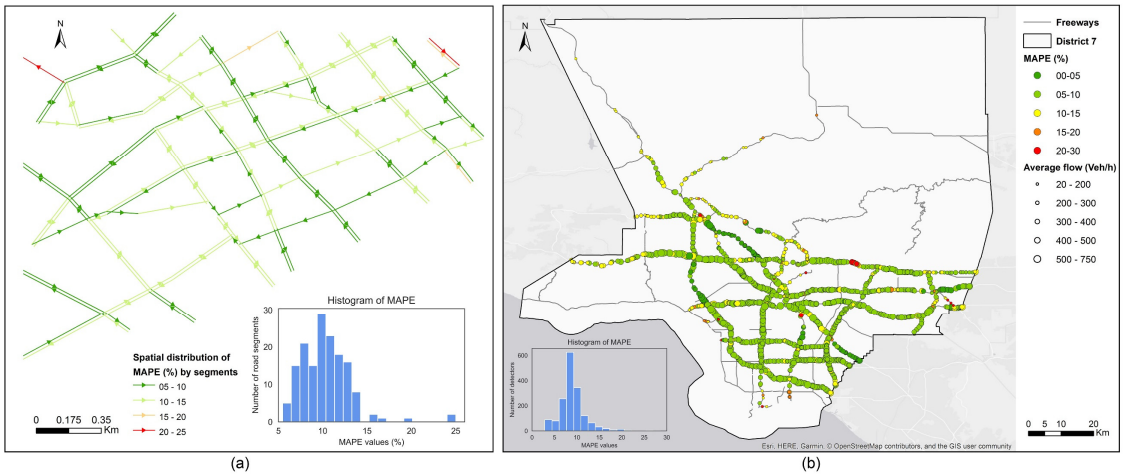

Fig 10. Spatial distribution of MAPE. (a) MAPE by road segments in SHSpeed traffic speed forecasting; (b) MAPE by detectors in PeMS traffic flow prediction. The colour of each segment/detector indicates the level of MAPE, and the size of each detector in (b) implies the level of average traffic flow. The histogram of MAPE by segment/detector is also presented in the two plots.

4.1.2 Prediction Error Distributions in Time

Fig 11 displays the distribution of the RMSE, MAE, and MAPE by the time of day on the SHSpeed and the PeMS data. In Fig (a)-(c) and (e)-(g), it shows the absolute errors, i.e., RMSEs and MAEs, vary with the average traffic speed/traffic flow. This is because both

MAE and RMSE consider only the magnitude of the deviations of predicted values from the observed values.

Alternatively, MAPE (i.e., relative errors) provides a better sense of prediction accuracy, as the errors examine the percentage deviations from the observations. According to Fig 11 (d) and (h), the most interesting finding is that the MAPE values during the daytime (i.e., rush hours 9:00-19:00) are lower than at night (i.e. off-peak hours). Thus, the proposed model generates better predictions in low traffic speed or high traffic flow situations (likely indicating congestions) than in high traffic speed or low traffic flow situations (usually indicating normal traffic conditions). In real-world applications, the forecast of traffic is more important and needed during peak-hours. For example, an accurate and reliable forecast of the traffic congestions with low speed and high flow can be used for real-time navigation to avoid delays. Hence, it can be said that the proposed method could be used to manage traffic proactively.

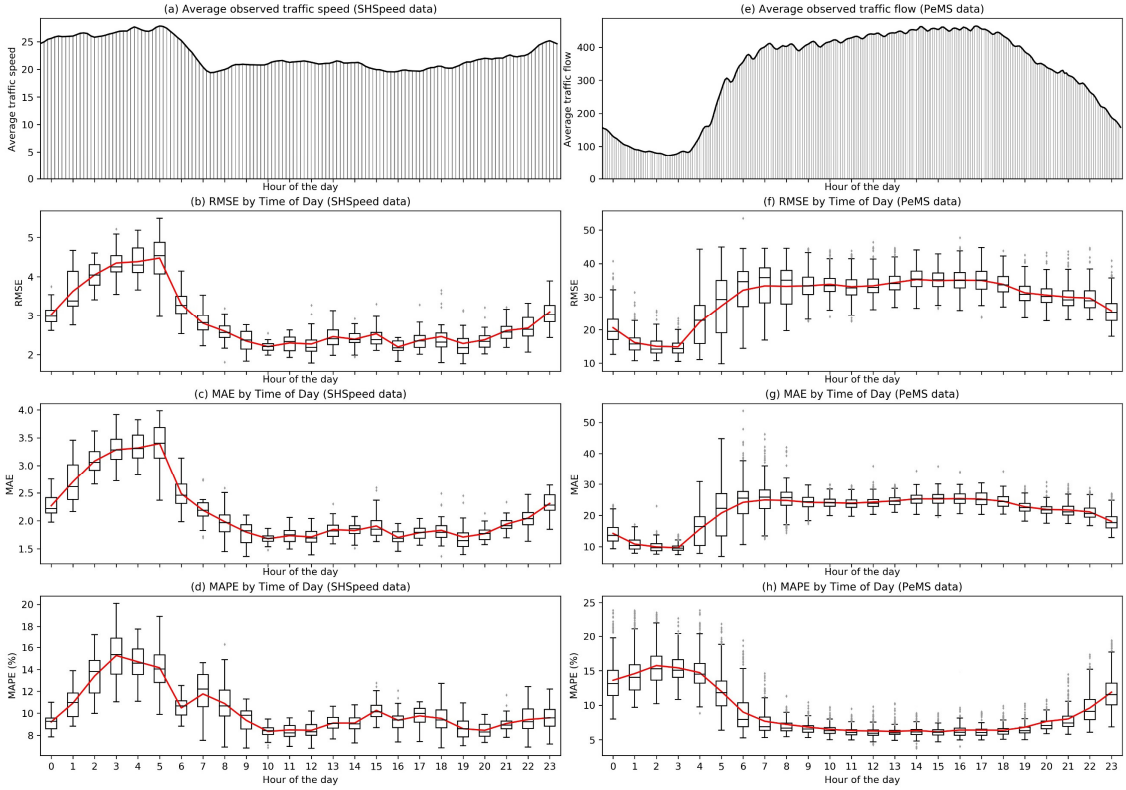

Fig 11. The distributions of the average observations, RMSE, MAE, and MAPE by time of day on SHSpeed data (subplots (a)-(d)) and PeMS data (subplots (e)-(h)). The red line represents the mean of the errors.

Fig 12 visualises the prediction error distributions over the testing days in the two prediction tasks. Although there exist several outliers, the variations of the three metrics by day are very small. The prediction performance of the proposed model is quite stable both on weekdays and on weekends.

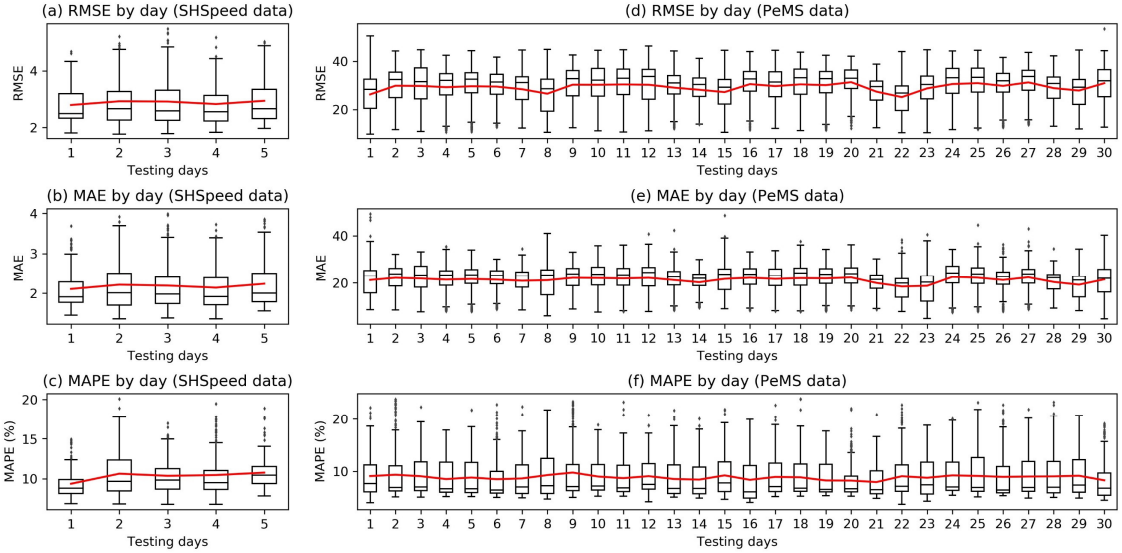

Fig 12. RMSE, MAE, and MAPE distributions over 5 testing days of SHSpeed data (subplots (a)-(c)) and over 30 testing days of PeMS data (subplots (d)-(f)). The red line represents the mean of the errors.

4.2 Sensitivity Analysis

In this section, a sensitivity analysis is conducted on several hyperparameters of the proposed model. First, a detailed discussion on the number of layers of RGC-LSTM is given. Additionally, from a spatial perspective, the influence of the order K of the graph convolution operator is analysed to reveal the influence of spatial span for spatial dependency modelling in short-term traffic forecasting. The impact of the time-window S, daily and weekly

temporal features are then investigated in regard to the prediction performance in the temporal domain.

4.2.1 Number of Layers

To explore the influence of the depth of RGC-LSTM on the prediction accuracy, we change the number of layers $\mathcal{L}$ from 1 to 4 while keeping the settings of other hyperparamters the same as given in Table 2 to test the forecasting error and runtime on both datasets. Results in Fig 13 show that the proposed model with two layers works best on SHSpeed data. In terms of PeMS data, the RMSE error continuously decreases with the increase of the number of layers. However, when increasing the depth of the model from 2 to 4, the accuracy improvement is non-significant while the runtime greatly increases. Thus, we choose the two-layer RGC-LSTM as the optimal configuration, as a trade-off between effectiveness and efficiency. The trends of the prediction errors are different in the two cases. It might be because that the PeMS dataset covers larger areas and more complex traffic patterns than SHSpeed dataset. Overall, the results indicate that the best number of layers of a DL model is not very large in traffic prediction tasks. Similar findings can be found in (Lv et al., 2015). Our results further confirm this.

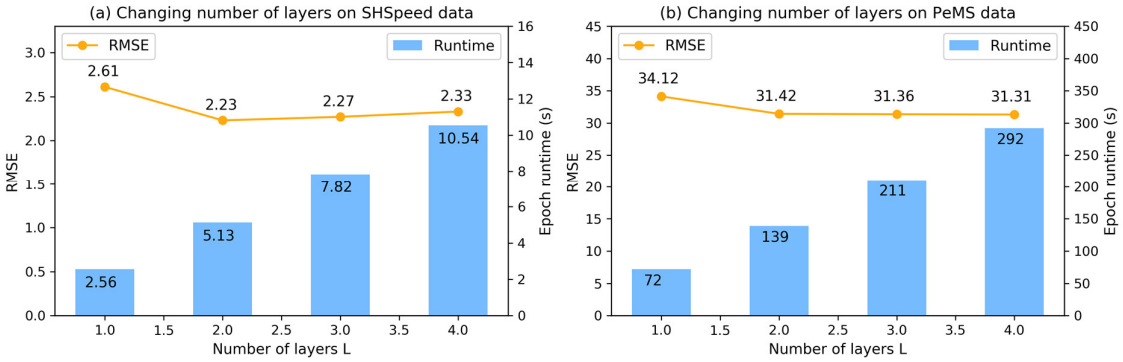

Fig 13. RMSE and epoch runtime change with the number of layers *L*.

4.2.2 *Spatial Dependency Analysis*

The graph convolution filter captures the spatial dependency of the traffic flow alongside the network. A higher order K of the graph convolution filter indicates a wider spatial span for spatial dependency modelling. To evaluate the impact of the spatial span on spatial dependency modelling, a sensitivity analysis on the order K is presented in this section.

According to Fig 14, it shows a larger K leads to a higher computational cost due to the increasing number of trainable variables in the model. The optimal K values for traffic speed and flow forecasting are two and three, respectively, which means the spatial span for spatial dependency modelling of an urban road network is smaller than that of the detector network. This may be because the spatial dependency between freeway detectors are stronger than between urban road segments. After increasing K to five, the RMSE is even larger than when ignoring the spatial dependency (i.e., let K=1). This is probably because too large a spatial span may include too much noise in the model, reducing its performance.

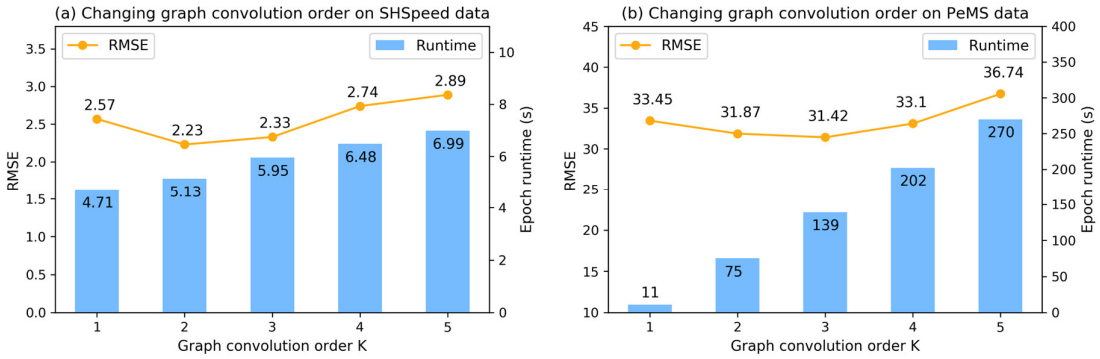

Fig 14. RMSE and epoch runtime change with graph convolution order K.

4.2.3 *Temporal Dependency Analysis*

Regarding temporal dependency modelling, the influence of the number of near-term features is first investigated. The proposed model is trained with different time-windows while the other parameters remain unchanged as stated in Section 3.2. Note that increasing the time-window does not change the number of variables in the model but increases the internal loops

in each RGC-LSTM layer. Fig 15 displays how the RMSE and the epoch runtime change with an increasing time-window in the two cases. It shows that the runtime of each epoch approximately increases by one second and 45 seconds for each additional near-term feature in the two tasks. In 10-min traffic speed forecasting, the RMSE achieves the minimum when *S* is four. In 5-min traffic flow prediction, the RMSE first considerably drops from 38.91 to the minimum (31.42) as the time window increases. There is then a gentle fluctuation in RMSE from an *S* of three to five, but the increase in training time is notable. It might produce a lower RMSE if we further enlarge the time window; however, this is not cost-effective considering the extremely long training time.

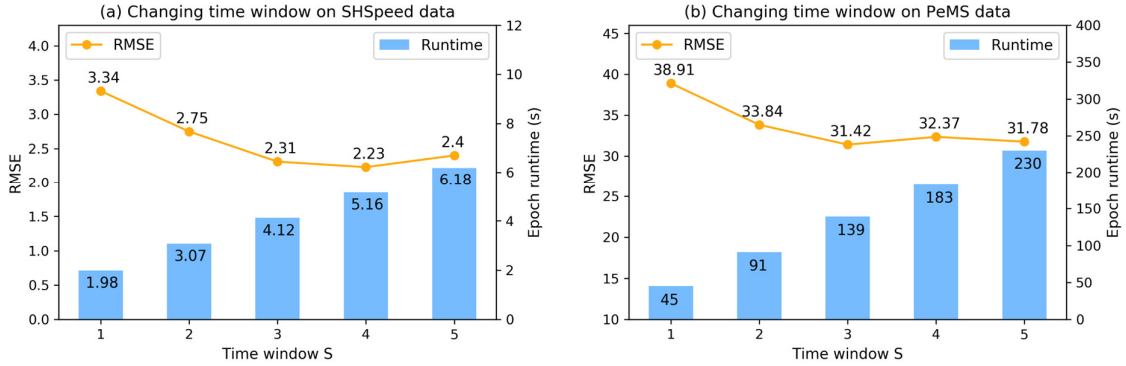

Fig 15. RMSE and epoch runtime change with time window *S*.

In addition, the effectiveness of fusing multiple periodic features for traffic forecasting is evaluated. Taking 5-min traffic flow forecasting as an example, results are given in Table 5. It shows that periodic features are contributing factors for short-term traffic prediction. Comparing the second and third rows, a very interesting finding is that the weekly feature contributes slightly more than the daily feature. Previous studies have reported that daily traffic patterns did vary from day to day to some extent and a one-week lag should be considered for more stable traffic prediction (Smith et al., 2002). Our finding also verifies

this lesson. In summary, the experiments show that fusing periodic features yields better prediction results.

Table 5. Evaluate the effectiveness of fusing daily and weekly features for short-term traffic forecasting.

Feature		RMSE	MAE	MAPE (%)
Daily	Weekly			
Yes	Yes	31.42	21.58	9.33
No	Yes	31.99	22.01	9.57
Yes	No	32.23	22.73	9.68
No	No	35.04	23.85	11.10

5 Conclusions and Future Work

This paper presents a spatial-temporal deep learning framework, termed RGC-LSTM, to forecast short-term traffic on large-scale networks. In this framework, a traffic network is represented as a graph $\mathcal{G} = (V, \varepsilon, A)$ with nodes as prediction locations, and edges indicating the adjacency relationship. The proposed RGC-LSTM is a stacking of multiple RGC-LSTM layers and a fully connected layer. In each RGC-LSTM layer, spatial dependencies on networks are captured by the proposed graph convolution operators and temporal features are extracted from a residual LSTM structure. The residual learning allows the model to go deeper and converge faster. In addition, the number of trainable parameters in the proposed model is independent of the size of the traffic network, and the computational cost is only of $\mathcal{O}(K|\varepsilon|)$ complexity, which is critical for the practical application of the model.

The proposed model is evaluated on a traffic speed dataset containing 156 road segments in Shanghai, China and a PeMS traffic flow dataset consisting of 1681 loop detectors deployed in District 7, California. The experiments demonstrate that the RGC-LSTM network is superior to many state-of-the-art baselines in terms of RMSE, MAE, and MAPE. The proposed model generates more accurate predictions during rush hours. We also discuss the spatial and temporal distribution of the prediction errors produced by RGC-LSTM and conduct the sensitivity analysis in space and time. It shows the performance of the

proposed model is consistent across different datasets both in time and space. The results of this effort provide new insights into the spatio-temporal modelling of network-based traffic data. For example, the weekly periodic feature is a more important contributing factor than the daily periodic feature in short-term traffic forecasting. In addition, the spatial span for spatial dependency modelling of urban road networks must not be too large.

However, this work still requires some improvements. First, missing data are a very common in many forecasting tasks, especially in the citywide traffic predictions. Future research should consider the missing data issues. Second, the model should be tested on other spatial-temporal forecasting tasks, such as road-network-based travel demand prediction, to further validate its effectiveness as a general framework. In addition, future research should further consider developing robust prediction models in the context of abnormal conditions (e.g., extreme weather and accidents), which is still an open problem and is attracting more attention. A possible way to take account these events into the DL model is via information fusion.

Acknowledgment

This work is part of the Consumer Data Research Centre (CDRC) project supported by the UK Economic and Social Research Council (ES/L011840/1). The first author's PhD research is jointly funded by China Scholarship Council (Grant No. 201603170309) and the Dean's Prize from University College London.

The authors would like to thank the editors Dr. Mey Yuan and Prof. Robert Weibel, and the anonymous referees for their constructive comments to improve the quality of the article. They are also grateful to Dr. James Haworth for many valuable discussions about the paper and related work.

Reference

Abadi, M., et al. TensorFlow: A System for Large-Scale Machine Learning. 12th Symposium on Operating Systems Design and Implementation (OSDI), 2016. 265-283.

Box, G. E. & Pierce, D. A. 1970. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association,* 65**,** 1509-1526.

Cai, P., et al. 2016. A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting. *Transportation Research Part C: Emerging Technologies,* 62**,** 21-34. https://doi.org/10.1016/j.trc.2015.11.002.

Castro-Neto, M., et al. 2009. Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert systems with applications,* 36**,** 6164-6173.

Chen, J., et al. 2018. Fine-grained prediction of urban population using mobile phone location data. *International Journal of Geographical Information Science,* 32**,** 1770-1786.

Chen, P., et al. 2016. Short-Term Traffic States Forecasting Considering Spatial–Temporal Impact on an Urban Expressway. *Transportation Research Record,* 2594**,** 61-72. https://doi.org/10.3141/2594-10.

Cheng, T., et al. 2012. Spatio-temporal autocorrelation of road network data. *Journal of Geographical Systems,* 14**,** 389-413.

Cheng, T., et al. 2014. A dynamic spatial weight matrix and localized space–time autoregressive integrated moving average for network modeling. *Geographical Analysis,* 46**,** 75-97.

Cheng, X., et al. Deeptransport: Learning spatial-temporal dependency for traffic condition forecasting. 2018 International Joint Conference on Neural Networks (IJCNN), 2018. IEEE, 1-8.

Cui, Z., et al. 2018. High-Order Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting. *arXiv preprint arXiv:1802.07007*.

Defferrard, M., et al. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in Neural Information Processing Systems, 2016. 3844-3852.

Dia, H. 2001. An object-oriented neural network approach to short-term traffic forecasting. *European Journal of Operational Research,* 131**,** 253-261.

Ding, Q. Y., et al. Forecasting traffic volume with space-time ARIMA model. Advanced Materials Research, 2011. Trans Tech Publ, 979-983.

Feng, X., et al. 2018. Adaptive Multi-Kernel SVM With Spatial-Temporal Correlation for Short-Term Traffic Flow Prediction. *IEEE Transactions on Intelligent Transportation Systems***,** 1-13. https://doi.org/10.1109/TITS.2018.2854913.

Graves, A. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Guo, J., et al. 2014. Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transportation Research Part C: Emerging Technologies,* 43**,** 50-64.

Haworth, J., et al. 2014. Local online kernel ridge regression for forecasting of urban travel times. *Transportation Research Part C: Emerging Technologies,* 46**,** 151-178. https://doi.org/10.1016/j.trc.2014.05.015.

He, K., et al. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016. 770-778.

Ke, J., et al. 2017. Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. *Transportation Research Part C: Emerging Technologies,* 85**,** 591-608.

Kingma, D. P. & Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kumar, K., et al. 2013. Short term traffic flow prediction for a non urban highway using artificial neural network. *Procedia-Social and Behavioral Sciences,* 104**,** 755-764.

Li, Y. & Shahabi, C. 2018. A brief overview of machine learning methods for short-term traffic forecasting and future directions. *SIGSPATIAL Special,* 10**,** 3-9.

Liao, S., et al. Large-scale short-term urban taxi demand forecasting using deep learning. Proceedings of the 23rd Asia and South Pacific Design Automation Conference, 2018. IEEE Press, 428-433.

Lin, L., et al. 2018. Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach. *Transportation Research Part C: Emerging Technologies,* 97**,** 258-276. https://doi.org/10.1016/j.trc.2018.10.011.

Liu, L., et al. 2018a. Attentive crowd flow machines. *arXiv preprint arXiv:1809.00101*.

Liu, Q., et al. 2018b. Short‐Term Traffic Speed Forecasting Based on Attention Convolutional Neural Network for Arterials. *Computer‐Aided Civil and Infrastructure Engineering,* 33**,** 999 − 1016.

Lv, Y., et al. 2015. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems,* 16**,** 865-873.

Ma, X., et al. 2017. Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction. *Sensors,* 17.

Min, X., et al. Short-term traffic flow forecasting of urban network based on dynamic STARIMA model. Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on, 2009. IEEE, 1-6.

Ren, Y., et al. 2019a. A hybrid integrated deep learning model for the prediction of citywide spatio-temporal flow volumes. *International Journal of Geographical Information Science*, 1-22. https://doi.org/10.1080/13658816.2019.1652303.

Ren, Y., et al. 2019b. Deep spatio-temporal residual neural networks for road-network-based data modeling. *International Journal of Geographical Information Science,* 33, 1894-1912. https://doi.org/10.1080/13658816.2019.1599895.

Shaw, S.-L., et al. 2016. Human dynamics in the mobile and big data era. *International Journal of Geographical Information Science,* 30, 1687-1693.

Shu, Y., et al. Wireless traffic modeling and prediction using seasonal ARIMA models. Communications, 2003. ICC'03. IEEE International Conference on, 2003. IEEE, 1675-1679.

Smith, B. L., et al. 2002. Comparison of parametric and nonparametric models for traffic flow forecasting. *Transportation Research Part C: Emerging Technologies,* 10, 303-321. https://doi.org/10.1016/S0968-090X(02)00009-8.

Sun, S., et al. 2006. A Bayesian network approach to traffic flow forecasting. *IEEE Transactions on intelligent transportation systems,* 7, 124-132.

Tian, Y. & Pan, L. Predicting short-term traffic flow by long short-term memory recurrent neural network. Smart City/SocialCom/SustainCom (SmartCity), 2015 IEEE International Conference on, 2015. IEEE, 153-158.

Van Lint, J., et al. 2002. Freeway travel time prediction with state-space neural networks: modeling state-space dynamics with recurrent neural networks. *Transportation Research Record: Journal of the Transportation Research Board*, 30-39.

Vlahogianni, E. I., et al. 2007. Spatio‐Temporal Short‐Term Urban Traffic Volume Forecasting Using Genetically Optimized Modular Networks. *Computer‐Aided Civil and Infrastructure Engineering,* 22, 317-325.

Vlahogianni, E. I., et al. 2014. Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies,* 43**,** 3-19. https://doi.org/10.1016/j.trc.2014.01.005.

Wang, X., et al. 2018. Efficient Metropolitan Traffic Prediction Based on Graph Recurrent Neural Network. *arXiv preprint arXiv:1811.00740*.

Wei, Y. & Chen, M.-C. 2012. Forecasting the short-term metro passenger flow with empirical mode decomposition and neural networks. *Transportation Research Part C: Emerging Technologies,* 21**,** 148-162.

Weisbrod, G., et al. 2003. Measuring economic costs of urban traffic congestion to business. *Transportation Research Record: Journal of the Transportation Research Board***,** 98-106.

Wu, Y. & Tan, H. 2016. Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. *arXiv preprint arXiv:1612.01022*.

Yang, Y., et al. 2017. Understanding structure of urban traffic network based on spatial-temporal correlation analysis. *Modern Physics Letters B,* 31**,** 1750230.

Yao, H., et al. 2018. Revisiting Spatial-Temporal Similarity: A Deep Learning Framework for Traffic Prediction. *arXiv preprint arXiv:1803.01254*.

Yu, B., et al. 2017a. Spatio-temporal Graph Convolutional Neural Network: A Deep Learning Framework for Traffic Forecasting. *arXiv preprint arXiv:1709.04875*.

Yu, G., et al. Short-term traffic flow forecasting based on Markov chain model. Intelligent Vehicles Symposium, 2003. Proceedings. IEEE, 2003. IEEE, 208-212.

Yu, H., et al. 2017b. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *Sensors,* 17**,** 1501.

Zhang, J., et al. DNN-based prediction model for spatio-temporal data. Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2016. ACM, 92.

Zhang, Y. & Cheng, T. 2019a. A Deep Learning Approach to Infer Employment Status of Passengers by Using Smart Card Data. *IEEE Transactions on Intelligent Transportation Systems*, 1-19. https://doi.org/10.1109/TITS.2019.2896460.

Zhang, Y. & Cheng, T. 2019b. Graph Deep Learning Model for Network-based Predictive Hotspot Mapping of Sparse Spatio-Temporal Events. *Computers, Environment and Urban Systems*, 1-12. https://doi.org/10.1016/j.compenvurbsys.2019.101403.

Zhang, Y., et al. 2019. A graph deep learning method for short-term traffic forecasting on large road networks. *Computer-Aided Civil and Infrastructure Engineering,* 34, 877–896. https://doi.org/10.1111/mice.12450.

Zhao, Z., et al. 2017. LSTM network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems,* 11, 68-75.