Mobile Data Traffic Prediction by Exploiting Time-Evolving User Mobility Patterns

Feiyang Sun, Pinghui Wang, Junzhou Zhao, Nuo Xu, Juxiang Zeng, Jing Tao, Kaikai Song, Chao Deng, John C.S. Lui, *Fellow, IEEE*, and Xiaohong Guan, *Fellow, IEEE*

Abstract—Understanding mobile data traffic and forecasting future traffic trend is beneficial to wireless carriers and service providers who need to perform resource allocation and energy saving management. However, predicting wireless traffic accurately at large-scale and fine-granularity is particularly challenging due to the following two factors: the spatial correlations between the network units (i.e., a cell tower or an access point) introduced by user arbitrary movements, and the time-evolving nature of user movements which frequently changes with time. In this paper, we use a time-evolving graph to formulate the time-evolving nature of user movements, and propose a model Graph-based Temporal Convolutional Network (GTCN) to predict the future traffic of each network unit in a wireless network. GTCN can bring significant benefits to two aspects. (1) GTCN can effectively learn intra- and inter-time spatial correlations between network units in a time-evolving graph through a node aggregation method. (2) GTCN can efficiently model the temporal dynamics of the mobile traffic trend from different network units through a temporal convolutional layer. Experimental results on two real-world datasets demonstrate the efficiency and efficacy of our method. Compared with state-of-the-art methods, the improvement of the prediction performance of our GTCN is 3.2 to 10.2 percent for different prediction horizons. GTCN also achieves 8.4× faster on prediction time.

Index Terms—User mobility, graph convolution, mobile computing, time series analysis

1 Introduction

With the popularity of smart mobile devices and the Internet of Things (IoT), mobile data traffic has sky-rocketed over the past few years. According to the technical report from Cisco [1], there will be more than 12 billion mobile devices and IoT connections, and the monthly global mobile data traffic will reach 77.5 exabytes (1 exabyte = 10^{18} bytes) per month by 2022. Globally, public WiFi hotspots (including public WiFi commercial hotspots and homespots) will grow four-fold from 124 million in 2017 to 549 million in 2022, and 59 percent of mobile data traffic will be offloaded to WiFi (compared to 54 percent in 2017). This trend will continue and create a huge market for commercial WiFi soon. However, the explosive growth of mobile data traffic and mobile users bring challenges to carry out network resource allocation and further impacts quality of service (QoS) while keeping high QoS for mobile network users is a significant issue for commercial WiFi services. To address these challenges, it is fundamental to model and understand traffic patterns of wireless networks [2], [3].

Predicting the future traffic of basic network units (e.g., wireless access points in WiFi networks and cell towers in cellular networks) is beneficial to mobile users, wireless carriers, and service providers who need to perform resource allocation, emergency events detection, and energy-saving management [3]. For example, network controllers can take preemptive actions for sudden increasings of traffic to avoid network congestion. It also reduces unnecessary operation cost by allocating energy and bandwidth based on future traffic demands. It has been studied and tested in deploying WiFi networks by Huawei and in deploying 5G networks by some companies like China Unicom [4]. Besides, precise mobile data traffic prediction is also beneficial for various applications. For instance, it can assist network administers to detect abnormal user behaviors and discover social events that may generate unexpectedly large data traffic [5]. Service providers can also set tiered pricing in time based on predicted traffic demands to ease network congestion and to achieve higher rewards [1], [6].

- *Feiyang Sun, Junzhou Zhao, Nuo Xu, and Juxiang Zeng are with the MOE KLINNS Laboratory, Xi'an Jiaotong University, Xi'an, Shaanxi 710054, China. E-mail: {fysun, nxu, jxzeng}@sei.xjtu.edu.cn, junzhou.zhao@xjtu.edu.cn.*
- *Pinghui Wang is with the Shenzhen Research Institute of Xi'an Jiaotong University, Shenzhen, Guangdong 518055, China, and also with the MOE KLINNS Laboratory, Xi'an Jiaotong University, Xi'an, Shaanxi 710054, China. E-mail: phwang@mail.xjtu.edu.cn.*
- *Jing Tao is with the MOE KLINNS Laboratory, Xi'an Jiaotong University, Xi'an, Shaanxi 710054, China. E-mail: jtao@mail.xjtu.edu.cn.*
- *Jing Tao is with the Shenzhen Research Institute of Xi'an Jiaotong University, Shenzhen, Guangdong 518055, China, and also with the Zhejiang Research Institute of Xi'an Jiaotong University, Hangzhou, Zhejiang 311215, China. E-mail: jtao@mail.xjtu.edu.cn.*
- *Kaikai Song is with the Huawei Noah's Ark Lab, Xi'an, Shaanxi 710400, China. E-mail: caesar.song@huawei.com.*
- *Chao Deng is with the China Mobile Research Institute, Beijing 100094, China. E-mail: dengchao@chinamobile.com.*
- *John C.S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. E-mail: cslui@cse.cuhk.edu.hk.*
- *Xiaohong Guan is with the MOE KLINNS Laboratory, Xi'an Jiaotong University, Xi'an, Xi'an, Shaanxi 710054, China, and also with the Tsinghua National Lab for Information Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: xhguan@xjtu.edu.cn.*

Manuscript received 9 June 2020; revised 12 Apr. 2021; accepted 30 Apr. 2021. Date of publication 11 May 2021; date of current version 3 Nov. 2022. (Corresponding author: Pinghui Wang.) Digital Object Identifier no. 10.1109/TMC.2021.3079117

1536-1233 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.
Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.

There has been a significant amount of effort by researchers on mobile data traffic prediction [7], [8], [9], [10], [11], [12], [13], [14], [15]. Besides time series prediction methods such as Seasonal AutoRegression Integrated Moving Average (SARIMA) [7] and Support Vector Regression (SVR) [8], recently, several works [10], [11], [13] adapt recurrent neural networks (RNNs) [16] and their variants to capture the temporal dynamics within the traffic series, which improves the accuracy of traffic prediction. For WiFi networks and cellular networks, mobile users typically share the same network interests and behaviors as others in nearby areas, so spatially close network units usually exhibit similar traffic trends. To capture such spatial correlation, several works [12], [15] formulate the network units as grid maps and propose models based on convolutional neural network (CNN) [17] to learn the spatial relations between the network units. Meanwhile, mobile users in areas with the same function share similar network behaviors, e.g., users often surf the Internet while waiting for the bus at bus stations. To capture such semantic correlation, several studies [13], [14], [18] introduce graph-based neural networks into wireless network modeling. They formulate the wireless network as a graph, where each node denotes a network unit and edges connect nodes that have the same function.

Despite the progress made, there remain some limitations:

- *(l1) Mobile users' mobility patterns are not exploited.* Existing works focus on exploiting explicit spatial relations such as distance between network units, but neglect implicit relations introduced by users' spatial-temporal mobility patterns. For instance, two cellular base stations that are far away from each other may have closely related traffic trends. The reason is that the base stations are located in two places connected by the subway and many people commute between these two places [19].
- *(l2) The variability of the wireless network units is not considered.*Existing works assume the network units of the wireless network are unchanging over time. However, many events can change it, e.g., regular maintenance of network facilities may turn off some nodes. Previous methods are unable to learn new features from the changed network unless they train the model on the entire network each time the network topology changes.

To develop an effective model that overcomes all the above limitations, the key challenges are threefold:

- *How to model user mobility considering its evolution over time?*The traditional way that formulates user trajectories as a series lacks expression for modeling spatial information. The proposed formulation should explicitly represent how user mobility changes over time.
- *How to jointly exploit user mobility patterns and the historical temporal traffic trend to forecast future traffic volumes?* The spatial information and temporal traffic are heterogeneous. The spatial information is usually formulated as grid maps or graphs, while the temporal traffic is naturally represented as time series. The desired method should integrate them into one model.

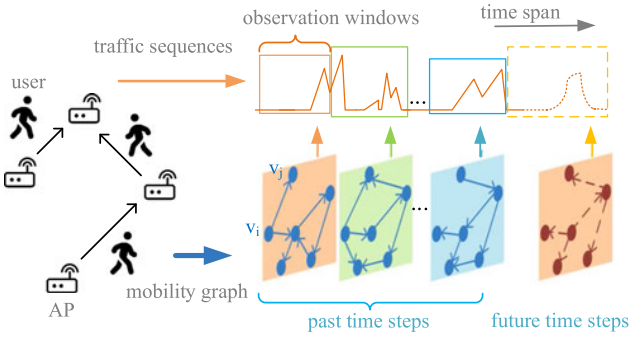


Fig. 1. An example of a user mobility graph. User mobility graph is a time-evolving graph, where each node denotes a network unit, i.e., a wireless access point (AP) or a cellular tower, and a directed edge $v_i \rightarrow v_j$ indicates that users are moving from v_i to v_j during the time step of interest. At each time step, we observe an r-length traffic volume sequence for each network unit. Given the observed traffic sequences and graph snapshots in the past few time steps, we aim to forecast the traffic trend of each network unit in future time steps. The network units provide data as inputs, and we do not require users to provide any additional information.

- *How to capture longer-term temporal dynamics of traffic sequences from various network units?* Accepting large-range input sequences helps us to capture more temporal features, which is beneficial to achieve better prediction performance. However, a larger input linearly increases the computational burden.

In this paper, we propose a traffic forecast model *Graph-based Temporal Convolutional Network* (GTCN) to address the above challenges. Specifically, to address the $c1$ challenge, we formulate the user mobility as an evolving graph as shown in Fig. 1, where each node represents a network unit and each edge describes a transfer behavior of users between network units at each time step. We propose a node aggregation scheme, which leverages user mobility trajectories to dynamically update node embeddings over time. To address the $c2$ challenge, we regard a node's historical traffic loads as its embedding in the time-evolving graph at the first aggregation step. Then, the node aggregation scheme adjusts the embedding to each node by aggregating both *intra-time relations* and *inter-time relations* among nodes (see Section 4.2). At last, the adjusted embeddings are used for temporal traffic trend prediction. To address the $c3$ challenge, we use temporal convolutional layers [20], which maintains exponential information by linearly stacking multiple layers and has reported better performance on many time series prediction tasks than RNN models [21]. We modify the temporal convolutional layer and use two types of convolution operations to distinguish the *individual traffic trend* of each node and the *common traffic trend* of all nodes (see Section 4.3). At the last regression layer, we compute the future traffic of each node. Because of the privacy and business issues, so far as we know, there is no large-scale mobile data traffic dataset publicly available. We collect mobile traffic datasets from WiFi networks in two university campuses and conduct our experiments on these large-scale real-world datasets.

Our contributions are summarized as follows:

1) We propose a deep learning model GTCN to predict mobile data traffic. To the best of our knowledge,

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.

GTCN is the first model that considers user mobility as evolving spatial information for temporal mobile traffic forecast.
2) We propose a node aggregation scheme, which learns the mobility patterns and also models how these patterns evolve over time.
3) We modify the temporal convolutional layer to enable the sharing of the common temporal trend from multiple input sequences.
4) We collect real-world large-scale mobile data traffic datasets and conduct our experiments on them. We will release the datasets to the public to foster reproducibility and research on this topic.

The rest of this paper is organized as follows. Section 2 summarizes related work. Section 3 describes our dataset and deployment and then defines our problem, and then Section 4 presents our GTCN model that exploits users' movement patterns for predicting network traffic. Section 5 makes some measurements and then presents the performance evaluation and testing results. Concluding remarks then follow in Section 6.

2 Related Work

Mobile Data Traffic Forecast. Considerable efforts have been devoted to model traffic in communication networks. [7], [8] show that traditional time series modeling algorithms such as seasonal ARIMA models and SVR models can be easily adapted to model and predict short-term wireless traffic load. [22], [23], [24] focus on predicting traffic load on each base station of cellular networks using probabilistic methods. However, as the forecasting window expanding, the predictive accuracy decreases rapidly. In recent years, with the popularity of location-based applications, exploring the correlation between traffic load and spatial characters attracts a lot of attention. Several spatial modeling methods have been proposed for traffic load in wireless networks [9]. Wang *et al.* [10] propose a deep learning model for traffic prediction in cellular networks. They use AutoEncoders [25] to learn spatial features between cellular towers and make predictions through Long Short-Term Memory (LSTM) [26]. Feng *et al.* [11] propose an end-to-end method DeepTP, which models spatial and temporal dependencies using a seq2seq model with an attention mechanism. These methods use a grid-based region partition, which is not suitable for most WiFi network management. The reason is that the grid-based partition considers each grid cell equally and unable to distinguish between different wireless access points (APs) in one grid cell, while APs are usually concentrated in a few grid cells (e.g., buildings). Wang *et al.* [14] study people transferring between regions and analyze their consumed network traffic. Wang *et al.* split the mobile traffic into inter-tower and intra-tower traffic, and then predict the future traffic of a cellular tower by aggregating the inter-tower traffic of its neighbors. They propose a graph neural network (GNN) method to capture such traffic patterns. However, their method has several limitations. First, they lack to consider the movement patterns of mobile users, which is important to explain why two distant places have similar traffic trends [19]. Second, they ignore that the wireless network structure can change over time because of the regular maintenance of groups of network facilities. Once the network structure changes, their model has to be retrained. In our work, we use a time-evolving graph, which dynamically records users' spatial-temporal movement, to capture the spatial correlations and temporal evolving dynamics between network units, which are exploited to improve the accuracy of network traffic prediction.

Spatio-Temporal Traffic Forecast. To some extent, mobile data traffic prediction tasks and road traffic prediction tasks have the same formulation. Specifically, existing mobile data traffic prediction studies [13], [14] (resp. road traffic prediction researches [27], [28], [29], [30]) formulate the wireless network (resp. road network) as a *graph*, where each node represents a cellular tower (resp. a monitor on a road section), each edge represents spatial relations between cellular towers (resp. intersections between road sections), and aiming to predict the future value of the mobile data traffic (resp. average traffic speed, or traffic flow) on each node. Therefore, we reference several state-of-the-art road traffic prediction methods. We also apply some of these methods to our dataset and compare them with our method. Yu *et al.* [27] propose a novel deep learning framework STGCN for traffic prediction. STGCN leverages graph convolution and gated temporal convolution, and combine them into a spatio-temporal convolutional block. STGCN stacks two such blocks to capture both spatial and temporal features. Guo *et al.* [28] propose ASTGCN which combines the attention mechanism and the spatial-temporal convolution. To capture both the spatial and temporal characteristics of traffic data, ASTGCN applies graph convolutions in the spatial dimension and standard convolutions in the temporal dimension. Zhao *et al.* [29] combine the GCN [31] and the GRU [32]. They use a GCN layer to capture the graph's topological structure to obtain the spatial dependence. Behind the GCN layer is a GRU model, which is used to capture the temporal change of node attributes (e.g., a time series of traffic) in the graph. Wu *et al.* [30] propose GraphWaveNet for spatial-temporal graph modeling. GraphWaveNet captures the spatial dependency between nodes by performing graph convolution in an adaptive adjacency matrix. To be able to handle long temporal sequences, GraphWaveNet uses stacked dilated convolution operations to increase its receptive field. However, all these methods learn spatial relations through a static graph, where the graph structure does not change over time. There are some works aiming to capture features from time-evolving graphs [33], where the graph structure changes with time. Diao *et al.* [33] propose a model DGCNN which utilizes tensor decomposition for Laplacian matrix estimation in graph convolutional layers. DGCNN is designed for minor changes in the graph structure. One weakness of DGCNN is that when most nodes and edges change at a time step, it may not obtain an accurate estimation of the Laplacian matrix. There are also some influential works [34], [35], [36], and all of them introduce new deep neural network architectures for traffic prediction. Zhang *et al.* [34] first design a branch of residual convolutional units to model complex factors, such as inter-region traffic, events, and weather. Pan *et al.* [35] employ a sequence-to-sequence architecture, which contains a meta graph attention network and a meta RNN to capture both spatial and temporal correlations. Li

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.

TABLE 1
Statistics of the Datasets Used in Our Experiments

Dataset	MDT-sub	MDT-urb
Collection Duration	May 3, 2015 – April 18, 2016	
Time Interval	15 minutes	
Covered Users	60,297	111,676
Covered APs	2,879	7,513
Covered Area	0.07 km^2	0.4 km^2
Flow Records (APs)	4.7×10^6	7.7×10^6
Sparseness	0.9513	0.9208

The sparseness refers to the ratio of zero values in the corresponding entire dataset.

et al. [36] make the first try to design an effective Neural Architecture Search (NAS) for spatio-temporal prediction task. In our work, the proposed method captures the variance of the correlations between nodes in a time-evolving graph at each time step, and so achieves better prediction performance.

3 Preliminaries

In this section, we describe our dataset, introduce the deployment of a WiFi network monitoring system, and formally formulate the mobile traffic forecast problem.

3.1 Dataset Description

Now, we introduce the datasets used in this work. Our work strictly follows ethical guidelines, and users' identity information such as mobile devices' MAC (Media Access Control) addresses has been anonymized to avoid privacy concerns.

From May 3, 2015, to April 18, 2016, we polled all wireless access points (APs) in two campuses located in a suburb or in urban of a city in China regularly (every 15 minutes) and collected information (e.g., MAC addresses, traffic volumes by each MAC address) of their connected mobile devices via SNMP (Simple Network Management Protocol) [37], [38]. Then, we can generate two Mobile Data Traffic datasets, denoted as *MDT-sub* and *MDT-urb*, from these two campuses, respectively. Based on this dataset, we know two kinds of information: 1) whether a mobile device is connected to an AP during a time interval; 2) how many traffic volumes a mobile device generates during a time interval.

Time Interval Setting. Our dataset is collected at 15-minute intervals. In practical applications, the best time interval is the time for the user to walk through a network unit because it can generate the finest trajectories without too much redundancy. For instance, each AP in a WiFi network covers an area of around 100 m^2. The time interval can be set at around 15 seconds. For cellular base stations that cover an area of 1 km^2, we can set the time interval as around 25 minutes. A larger time interval will reduce the model performance because it models user movement roughly, but a smaller time interval brings a computational burden.

Statistics. MDT-sub and MDT-urb were collected on 2,879 and 7,513 wireless access points (APs), respectively. We aggregated the traffic every 15 minutes, so each AP contains 96 data points per day, and there are 33,792 data points for the entire collection time. For the suburb campus with a gross floor area of 0.07 km^2 and the urban campus with a gross floor area of 0.4 km^2, each AP covers an area of about 100 m^2 in buildings. During the one year, there exist 60,297 and 111,676 active mobile devices on the Internet for the suburb and the urban campuses respectively. On average, a mobile device was 196 (resp. 189) days active and connected to 76 (resp. 113) APs in the suburb campus (resp. the urban campus), respectively. Table 1 shows more details about our datasets. Some measurements and illustrations can be found in Section 5.5 and Fig. 9.

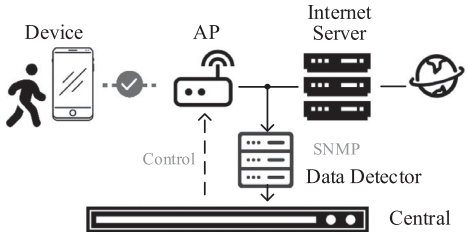


Fig. 2. Deployment in a WiFi network.

Normalization. Mobile data traffic volumes have a wide range from 0 packets to 10^8 packets and approximately follow a power-law distribution, where large traffic only accounts for a very small fraction. It is important to normalize the traffic volumes [39] because large data will lead to a great leap in the gradient landscape in the backpropagation of neural networks, which reduces the stability of the training process. Therefore, we normalize the traffic into a small interval. Specifically, we use a $\log_{10}$ function to zoom the traffic volumes $\mathbf{Y}$ to a small scale, as follows: $\mathbf{Y}_{\text{norm}} = \log_{\mathbf{10}}\mathbf{Y}$.

3.2 Deployment

Before we introduce the problem we focus on, let us discuss how to deploy our model and what benefits this model brings.

Deployment. Beyond a WiFi network, we deploy a Data Detector to collect the traffic volumes and devices' connection records via SNMP (see Fig. 2). The Detector outputs the users' movement matrix and APs' historical traffic volumes (which will be detailed in the next subsection) as the inputs of our prediction model. To generate the users' movement matrix, the Detector needs to record the APs that a user (i.e., MAC address) connected to at the previous time interval. To avoid privacy concerns in practical applications, we can anonymize the MAC addresses via hash techniques like SHA-3 (Secure Hash Algorithm 3) [40], which turns a text into a short hash value. SHA-3 provides a secure one-way function which means we can not reconstruct the input MAC from the hash output. Then, the Central server is used to deploy our model that is detailed in Section 4. The Central server predicts the future traffic and delivers control commands to the network units.

Incentive. Traffic prediction is beneficial for many applications, such as resource allocation, emergency events detection, and energy-saving management. We provide an experimental example in Section 5.6. The result shows significant improvement in energy-saving while the deployment cost is minor.

Apply to Other Types of Wireless Networks. Consider that the mobility pattern of 4G/5G users may differ from Wi-Fi users because both their average movement speeds and

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.

their average distances between network units are different. We should adjust the time interval in our model according to the application scenario.

3.3 Problem Formulation

Now, we formally describe the mobile traffic forecast problem. In a wireless network, given the mobile data traffic histories of network units and mobile users' mobility trajectories, our goal is to predict the mobile data traffic of network units in future time steps. Before we formally define the problem, we first introduce some notations.

We denote a set of nodes as $\mathcal{V} \triangleq \{v_1, v_2, \ldots, v_N\}$, where each node represents a network unit (e.g., an AP) in the wireless network of interest. At any time step t, we use an adjacency matrix $A^{(t)} \in \{0,1\}^{N \times N}$ to describe the node connections. The (i,j)-th entry $A_{ij}^{(t)} = 1$ if there is a directed edge from node v_j to node v_i at time step t, e.g., a user moves from node v_j to node v_i at t. Let $a_i^{(t)}$ denote the traffic volume of node v_i at time step t, and let $\mathbf{s}_i^{(t)} \triangleq [a_i^{(t-r+1)}, a_i^{(t-r+2)}, \ldots, a_i^{(t)}]^\top \in \mathbb{R}^r$ be an r-dimensional vector of the historical traffic volumes of node v_i in the past r time steps. Let $S^{(t)} \triangleq [\mathbf{s}_1^{(t)}, \ldots, \mathbf{s}_N^{(t)}] \in \mathbb{R}^{r \times N}$ be the historical traffic volume matrix of all nodes. The graph snapshot observed at time step t is denoted by $\mathcal{G}^{(t)} \triangleq (A^{(t)}, S^{(t)})$, which includes the information of both node connections and node traffic volumes. Let $S_L^{(t)} \in \mathbb{R}^{1 \times N}$ denote the last row (i.e., the rth row) of $S^{(t)}$, i.e., the traffic volume of all N nodes at time step t.

Given a sequence of graph snapshots $\{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(t)}\}$, we want to predict the future p-step traffic volume matrices $\{S_L^{(t+1)}, \ldots, S_L^{(t+p)}\}$.

Discussions. For traffic prediction tasks, it is significant to describe traffic changes between different APs (i.e., nodes). Therefore, we combine historical traffic (as node features) and binary adjacency matrix (describe node connections) to describe traffic changes between different APs. Here, we do not use a weighted adjacency matrix generated through the number of users in the APs because its performance is not better than the binary adjacency matrix. The prediction results using the weighted adjacency matrix or the binary adjacency matrix are almost the same.

4 Graph-based Temporal Convolution Network (GTCN)

In this section, we first introduce the basic framework of our proposed GTCN model. Then, we elaborate the two modules in GTCN to describe how to predict future traffic volume matrices.

4.1 Overview

The framework of GTCN is illustrated in Fig. 3. GTCN takes the graph snapshots $\{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(t)}\}$ as input, and outputs the predicted future node traffic volumes. There are two modules in GTCN, namely, the *node embedding module* and the *traffic prediction module*. The node embedding module leverages the adjacency matrices $\{A^{(1)}, \ldots, A^{(t)}\}$ to obtain node embeddings, which could well capture the connection/mobility relations among nodes. The traffic prediction module leverages the historical traffic volume matrices $\{S^{(1)}, \ldots, S^{(t)}\}$ and the previously obtained node embeddings to predict future traffic volume matrices.

4.2 Node Embedding

Recall that an adjacency matrix $A^{(t)}$ represents the connections among nodes (e.g., APs in a wireless network). These connections actually capture the mobility patterns of users, as $A_{ij}^{(t)} = 1$ means that there are users moving from node v_j to v_i at time step t. Such mobility relations could reflect traffic volume relevance among nodes. For example, if users generated large traffic volumes at node v_j, then after the users moved to node v_i, they are also likely to generate large traffic volumes at node v_i. Therefore, traffic volume information of nodes are correlated via adjacency matrix $A^{(t)}$.

In addition, the adjacency matrix is evolving, from $A^{(1)}$ at time step 1 to $A^{(t)}$ at time step t. Each element $A_{ij}^{(t)}$ may change to 1 (or 0) at the next time step if there are users (or no users) moving from node v_j to v_i at the next time step, as users at node v_j can choose to stay or leave. Such mobility patterns could reflect nodes' traffic trends. For example, if most users pass through node v_j, v_j's traffic volume will change frequently. If users tend to stay at v_j, v_j's traffic volume will be relatively stable. Therefore, modeling how users' mobility patterns (staying or leaving) evolve is also useful for traffic prediction.

In this subsection, we study above two mobility relations. We call the previous one as *intra-time relations*, as it models user movements between nodes at each time step. We call the later one as *inter-time relations*, as it models how user mobility patterns at different nodes change with time.

4.2.1 Modeling Intra-Time Relations

At a particular time step, we aggregate traffic volume information of each node's neighbors in order to capture intra-time relations among nodes.

Consider the user mobility graph at time t, i.e., $\mathcal{G}_t = (A^{(t)}, S^{(t)})$. Note that $A_{ij}^{(t)} = 1$ if a user moves from node v_j to node v_i at t. Let $\mathcal{N}(v_i)$ denote the set of in-neighbors of node v_i in $\mathcal{G}_t$. We use a vector (or, embedding) $\mathbf{h}_{v_i} \in \mathbb{R}^r$ to represent the traffic volume information of node v_i. Neighboring nodes' traffic volume information can be aggregated by

$$\mathbf{h}_{\mathcal{N}(v_i)} = \mathcal{A}\Big(\{v_j : v_j \in \mathcal{N}(v_i)\}\Big), \tag{1}$$

where $\mathcal{A}(\cdot)$ denotes an aggregation function. We consider to use different aggregation functions as follows:

- *Mean function.* The mean function simply averages embeddings of the node's neighbors, i.e.,

$$\mathcal{A}_{\text{mean}}\big(\mathcal{N}(v_i)\big) \triangleq \frac{1}{|\mathcal{N}(v_i)|} \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{h}_{v_j}.$$

- *Spectral convolution function.* We can use the spectral convolution to capture more information by

$$\mathcal{A}_{\text{conv}}\big(\mathcal{N}(v_i)\big) \triangleq \sigma\Big(\mathbf{W}_c \cdot \mathcal{A}_{\text{mean}}\big(\{v_i\} \cup \mathcal{N}(v_i)\big) + \mathbf{b}_c\Big),$$

where $\mathbf{W}_c$, $\mathbf{b}_c$ are trainable parameters and $\sigma(\cdot)$ is the sigmoid activation function. The traffic volumes of

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.

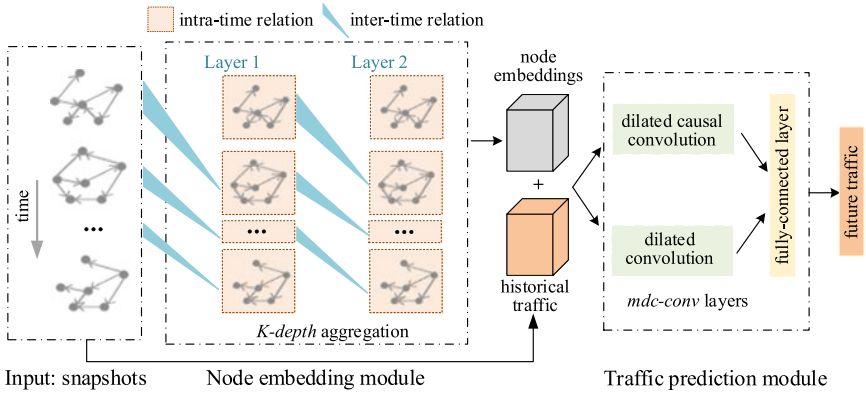


Fig. 3. Overview of our GTCN model.

node v_i and its neighbors are gathered by a mean operation, and then filtered by a parameter matrix. Such operation is a linear approximation of a localized spectral convolution [31], i.e., a convolutional kernel $\mathbf{W}_c$ is performed on each node $v_i \in \mathcal{V}$ and its neighborhood $\mathcal{N}(v_i)$ to generate a new embedding.

- *Pooling function*. The spectral convolution function aggregates information of all neighbors' traffic volumes into the new embedding. Sometimes a node may be only relevant to a few neighbors. To avoid influence from the irrelevant nodes, a pooling operation is applied to capture which features contribute more to the new embedding. In the pooling operation, we input node v_i's neighbors into a fully-connected neural layer, and then apply an element-wise max-pooling, i.e.,

$$\mathcal{A}_{\text{pool}}\big(\mathcal{N}(v_i)\big) \triangleq \max\big\{\sigma(\mathbf{W}_p\mathbf{h}_{v_j} + \mathbf{b}_p) : v_j \in \mathcal{N}(v_i)\big\},$$

where $\max\{\cdot\}$ denotes the element-wise max operation, $\mathbf{W}_p$ and $\mathbf{b}_p$ are the trainable parameters of this function.

In our experiments, we choose both convolutional and pooling functions to calculate $\mathcal{N}(v_i)$'s embedding and then concatenate them together as the final embedding $\mathbf{h}_{\mathcal{N}(v_i)} \in \mathbb{R}^{2r_a}$, where r_a is the output dimension of the aggregation functions.

4.2.2 Modeling Inter-Time Relations

Now we describe how to aggregate and update nodes' inter-time relations at different time steps. We propose *K-depth aggregation scheme* that uses K layers to aggregate traffic volumes over previous K time steps from snapshots $\mathcal{G}_{t-K+1}, \ldots, \mathcal{G}_t$. It explicitly models the evolving patterns of user mobility by imitating how users move between nodes. The scheme is illustrated in Fig. 4.

We denote the aggregated embedding of a node v_i in the kth layer at time t as $\mathbf{h}_{v_i}^{k,t} \in \mathbb{R}^{r_k}$, where r_k is the output dimension of the kth layer. The aggregation scheme is presented by

$$\underbrace{\mathbf{h}_{v_i}^{k,t}}_{\text{output}} = \sigma\Big(\mathbf{W}^k \cdot \big(\underbrace{\mathbf{h}_{v_i}^{k-1,t-1}}_{\text{inter-time}} \,\|\, \underbrace{\mathbf{h}_{\mathcal{N}(v_i)}^{k,t}}_{\text{intra-time}} \,\|\, \underbrace{s_i^{(t)}}_{\text{traffic}}\big) + \mathbf{b}^k\Big), \tag{2}$$

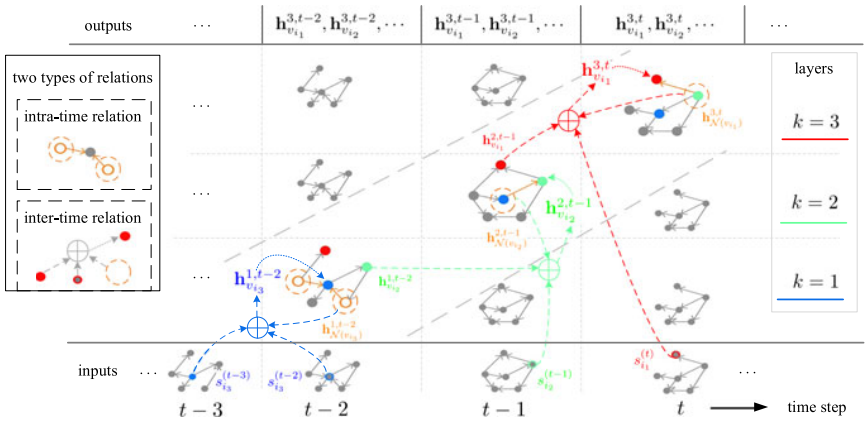
where $\|$ denotes the concatenation operation. The first input element $\mathbf{h}_{v_i}^{k-1,t-1} \in \mathbb{R}^{r_{k-1}}$ denotes the embedding of v_i calculated by the $(k-1)$-th layer using the $(t-1)$-th snapshot. It models nodes' inter-time relations between two adjacent time steps. The second input element $\mathbf{h}_{\mathcal{N}(v_i)}^{k,t} \in \mathbb{R}^{2r_a}$, which is obtained by the Eq. (1), models the intra-time relation between v_i and its neighbors at time t. The third input element is the initial embedding, i.e., the observed traffic sequence $s_i^{(t)} \in S^{(t)}$ of v_i at time t. Specifically, for the first layer, i.e., $k = 1$, the first input element is replaced by the initial embedding of each node. To transform the concatenated vector to $\mathbf{h}_{v_i}^{k,t}$, we input the concatenated vector into a fully connected layer, where $\mathbf{W}^k$ and $\mathbf{b}^k$ are trainable parameter matrices. The L_2 normalization is conducted for $\mathbf{h}_{v_i}^{k,t}$ later, i.e., $\mathbf{h}_{v_i}^{k,t} \leftarrow \mathbf{h}_{v_i}^{k,t}/\|\mathbf{h}_{v_i}^{k,t}\|_2$. The output of layer K is the final embedding, e.g., the final embedding of node v_i at time step t is denoted as $\mathbf{h}_{v_i}^{K,t}$. We combine all nodes' final embeddings at time t together and denote it as $\mathbf{h}_{\mathcal{V}}^{K,t} \in \mathbb{R}^{r \times N}$.

4.2.3 Discussions

Differences From GraphSage [41]. Our framework is inspired by an inductive graph convolutional framework GraphSage. The main difference is that GraphSage only aggregates information of connected nodes in the vertex domain (spatial domain), while our method also captures evolving patterns in the time-evolving user mobility graph beyond the spatial domain.

Advantages of Aggregation-Based Method Comparing With Spectral Convolutional Method GCN [31]. GCN uses full-batch gradient descent, which requires storing all the intermediate embeddings to compute the full gradient, leading to expensive memory requirements. Storing the intermediate embeddings for every time step in a time-evolving graph costs too much space. Our *K-depth* scheme is trained using mini-batch gradient [41]. It only stores a few subsets of nodes in each training batch to reduce the memory requirement. *Does the Representation $\mathbf{h}_{\mathcal{V}}^{K,t}$ Capture Enough Information for Modeling Temporal Dynamics of Mobile Data Traffic?*. The *K-depth* scheme regards traffic sequences as the initial embedding of nodes, and searches through previous K time steps to aggregate information to update nodes' embeddings. It implicitly models the temporal trend of mobile data traffic. We conduct experiment, which sets $K = 12$ for *K-depth* scheme, puts the $\mathbf{h}_{\mathcal{V}}^{K,t}$ into a fully connected layer, and uses the future traffic $S^{(t+1)}$ to supervise the training process. The prediction results do not outperform the

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.



K-depth aggregation scheme

Fig. 4. Overview of the K-depth aggregation scheme. For instance, here we use K-depth with $K=3$ to compute the embedding of the red node v_{i_1} at time t. In the third layer (i.e., $k=3$, the red part in the figure), the embedding $\mathbf{h}_{v_{i_1}}^{3,t}$ is calculated following the Equation (2) using three input elements: the previous embedding $\mathbf{h}_{v_{i_1}}^{2,t-1}$ of v_{i_1} at time $t-1$, the intra-time relations $\mathbf{h}_{\mathcal{N}(v_{i_1})}^{3,t}$ from v_{i_1}'s neighbors, and the initial embedding $s_{i_1}^{(t)}$ of v_{i_1}. The intra-time relations are calculated following the Equation (1) using the embedding $\mathbf{h}_{v_{i_2}}^{2,t-1}$ of $v_{i_2} \in \mathcal{N}(v_{i_1})$. The first layer and second layer follow the same process. In first layer, we specifically use the initial embedding at the previous time step of each node as the first input element. Finally, we model the evolving process from time $t-2$ to t and obtain embedding $\mathbf{h}_{v_{i_1}}^{3,t}$ of node v_{i_1}. We make this computation over each graph snapshot and the entire time span in parallel for acceleration.

comparing methods. Therefore, we still need a more powerful tool to model the temporal dynamics of mobile data traffic.

4.3 Modeling Long-Term Temporal Patterns

Note that the previously obtained node embeddings only consider inter-time relations between adjacent time steps. In addition, user mobility also has long-term temporal patterns. Mining such patterns can also help traffic prediction. For example, in the morning, people move from home to workplaces, while in the afternoon, people move from workplaces to home; thus traffic trends at workplace and home will influence each other. Different nodes have their own traffic patterns, referred to as the *individual trend*. Meanwhile, different nodes also have common traffic patterns, e.g., traffic volume of most nodes increases at rush hour in the morning, referred to as the *common trend*.

In order to model both trends, in this traffic prediction module, we propose the *multi-input dilated causal convolution* (mdc-conv), built on the *dilated causal convolution* [20] and *residual structure* [21], [42].

4.3.1 Multi-Input Dilated Causal Convolution

The causal convolution is a special case of one dimensional convolution, as it only looks back, i.e., an output at time t is convolved only with elements from time t and earlier in the previous layer. The dilated causal convolution (dc-conv) [20] extends the receptive field of causal convolutions by skipping elements with a certain dilation factor when strides over inputs. A simple example of the dc-conv is as follows: given a one dimensional sequence input $\mathbf{x} \in \mathbb{R}^n$ and a filter $f: \{0, \ldots, w-1\} \mapsto \mathbb{R}$, the dc-conv of $\mathbf{x}$ with f at tth element is defined as

$$\text{dc-conv}(\mathbf{x}, t) \triangleq (\mathbf{x} *_c f)(t) = \sum_{j=0}^{w-1} f(j) \cdot \mathbf{x}^{(t-d \cdot j)}, \tag{3}$$

where $*_c$ denotes the dilated causal convolution operation, d is the dilation factor, and w is the filter size. The filter multiplies previous w elements by its w parameters (or parameter vectors, or parameter matrices, depending on the dimension of input), and outputs a sequence having the same length as the input.

In order to capture the individual trend of multiple input sequences generated by different nodes, a general approach is to train independent dc-conv models for each node. However, this approach ignores to leverage the common trend of different nodes, e.g., traffic volume of most nodes increases at rush hour in the morning. We propose the mdc-conv operation to address this issue, as illustrated in Fig. 5.

The input of mdc-conv layers is a sequence with time span T, denoted by $\mathbf{X} \in \mathbb{R}^{2r \times N \times T}$, where each element $\mathbf{X}^{(t)} = \mathbf{h}_{\mathcal{V}}^{K,t} || S^{(t)} \in \mathbb{R}^{2r \times N}$ is a concatenation of the learned embedding $\mathbf{h}_{\mathcal{V}}^{K,t} \in \mathbb{R}^{r \times N}$ of all nodes $\mathcal{V}$ ($\|\mathcal{V}\| = N$) and traffic volume matrices $S^{(t)} \in \mathbb{R}^{r \times N}$. $\mathbf{X}^{(t)}$ is sliced by the time dimension of $\mathbf{X}$. It can be used to learn the individual trend of the input. To capture the common trend of different nodes, we also slice $\mathbf{X}$ by the node dimension, denoted by $\mathbf{X}_{(i)} \in \mathbb{R}^{2r \times T}$, where i is the index of node $v_i \in \mathcal{V}$.

The mdc-conv layer uses two types of convolution operations to process these two parts of inputs. We set two filters $f_1: \{0, \ldots, w-1\} \mapsto \mathbb{R}^{2r \times N}$ and $f_2: \{0, \ldots, w-1\} \mapsto \mathbb{R}^{2r \times T}$. Then, the mdc-conv operation at time t and index i is represented as

$$\begin{aligned} &\text{mdc-conv}\big((\mathbf{X}, t); (\mathbf{X}, i)\big) \\ &\triangleq \big\{ \underbrace{(\mathbf{X} *_c f_1)(t)}_{\text{slicedbytime}}; \underbrace{(\mathbf{X} * f_2)(i)}_{\text{slicedbynode}} \big\} \\ &= \left\{ \sum_{j=0}^{w-1} f_1(j) \odot \mathbf{X}^{(t-d_1 \cdot j)}; \sum_{j=0}^{w-1} f_2(j) \odot \mathbf{X}_{(i-d_2 \cdot j \cdot (-1)^j)} \right\}, \end{aligned} \tag{4}$$

where $\odot$ is the element-wise product, $*$ denotes the dilated convolution operation, and d_1, d_2 are the dilation factors. The first part of mdc-conv is similar to the dc-conv which captures the individual trend from a temporal sequence,

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.

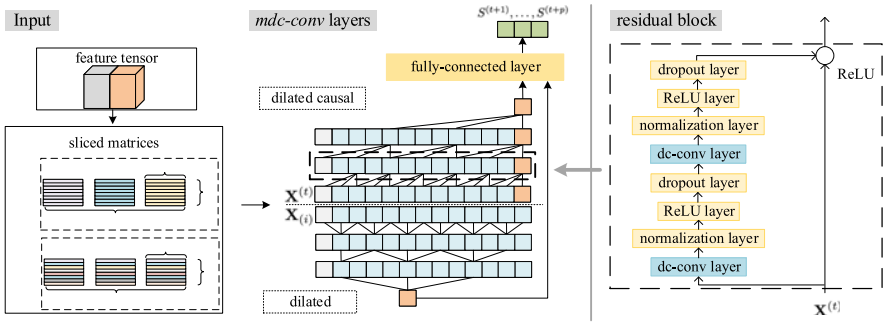


Fig. 5. Overview of the Temporal Dynamic Prediction module. An illusion of the dilated causal convolutions (left) and the structure of an alternative operation, residual block (right).

while the second part is willing to learn the common trend from different nodes. The input of the second part is a sequence of nodes, which are not temporally arranged. As we know if the sequence is in arbitrary order, the convolution operation may not capture meaningful features. To address this issue, the node sequence is expected to be arranged exploiting prior knowledge, e.g., if we cluster and arrange the nodes by distance, the dilated convolution could summarize the features of adjacent nodes at the first layer and expand its receptive field at subsequent layers.

We denote the two parts' outputs of several mdc-conv layers as $\mathbf{Z}_1$ and $\mathbf{Z}_2$. The mdc-conv outputs a sequence having the same dimension as the input. We can exponentially expand the effective receptive field of our model by stacking multiple mdc-conv layers, which enables our model to capture long-term temporal patterns with a few layers and save computation resources.

4.3.2 Residual Blocks

Our model is able to look through a long sequence by stacking many layers. Residual architectures use an identity mapping which transfers the features in shallow layers to deeper layers. It benefits the backpropagation of gradients, which has been demonstrated as a good solution for very deep neural networks in many applications [43]. We use $\mathcal{F}$ to denote a series of layers which stacked by two groups of a mdc-conv layer, a normalization layer, a rectified linear unit (ReLU) layer, and a dropout layer. The normalization layer is applied for accelerating convergence of the training process. The ReLU layer can relieve the vanishing gradient problem. The dropout layer benefits the regularization of parameters, which avoids overfitting. The output of a residual block is combined by an input $\mathbf{X}$ and the output $\mathcal{F}(\mathbf{X})$ of the series of layers, described as

$$\mathrm{ReLU}(\mathbf{X} \oplus \mathcal{F}(\mathbf{X})), \tag{5}$$

where $\oplus$ denotes element-wise adding and ReLU denotes a rectified linear unit operation. We employ residual blocks in place of the mdc-conv layers.

4.3.3 Discussions

How to Deal With Changes in the Network Topology. Our model can easily adapt to the deletion of nodes or the changes of edges in the graph. When adding new nodes, the node embedding module can generate embeddings of new nodes because it works by aggregating the neighbors of the node instead of the entire graph. However, the traffic prediction module is not suitable for new nodes because the input size is fixed after training. In this situation, we can extend the filters f_1 and fine-tune the traffic prediction module based on the existing model.

4.4 Traffic Prediction

After stacking several residual blocks, we obtain the final output $\mathbf{Z}_1 \in \mathbb{R}^{2r \times N}$ and $\mathbf{Z}_2 \in \mathbb{R}^{2r \times T}$. $\mathbf{Z}_1$ contains the individual trend of each node and $\mathbf{Z}_2$ contains the common trend of different nodes. In order to leverage the common trend learned in $\mathbf{Z}_2$ into each node, we multiply these two parts as $\mathbf{Z}_2^{\mathsf{T}} \cdot \mathbf{Z}_1$. Finally, we use a fully-connected layer to predict the future traffic volume $S^p \triangleq \{S_{r:}^{(t+1)}, \ldots, S_{r:}^{(t+p)}\}$ as

$$\widehat{S^p} = \mathrm{FCN}\big(\mathbf{Z}_1 || (\mathbf{Z}_2^{\mathsf{T}} \cdot \mathbf{Z}_1)\big), \tag{6}$$

where FCN refers to a fully-connected layer. In the training process, the goal is to minimize the error between $\widehat{S^p}$ and its real value S^p. Formally, the loss function of our model is

$$\mathcal{L} = \|S^p - \widehat{S^p}\|_2 + \lambda L_{reg}, \tag{7}$$

where the first term is used to minimize the error between the real traffic and the prediction, the second term L_{reg} is an L_2 regularization term that helps to avoid over-fitting, and λ is a hyperparameter.

5 Experiments

In this section, we conduct experiments on real-world datasets to answer the following research questions:

- *RQ1* What is the prediction performance of GTCN when comparing with other baselines?
- *RQ2*. What is the benefit of dynamically upgrading the feature of each node in the time-evolving graph?
- *RQ3*. What are the advantages for temporal convolutional networks when comparing with the RNN models?
- *RQ4*. What are the factors that impact the prediction performance on real-world datasets?
- *RQ5*. What applications can our model contribute to?

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.

5.1 Experimental Settings

5.1.1 Evaluation Metrics

To evaluate the accuracy of forecast traffic volumes $\widehat{\mathbf{Y}}$ with respect to the ground truth $\mathbf{Y}$, we use the following four measurement metrics:

- *Root Mean Squared Error (RMSE).*

$$\mathrm{RMSE} = \sqrt{\frac{1}{N \cdot T}\sum_{i=1}^{N \cdot T}(\mathbf{Y}_i - \widehat{\mathbf{Y}}_i)^2},$$

  where N is the number of network units in the testing data, and T is the time span of the testing data.
- *Mean Absolute Error (MAE).*

$$\mathrm{MAE} = \frac{1}{N \cdot T}\sum_{i=1}^{N \cdot T}|\mathbf{Y}_i - \widehat{\mathbf{Y}}_i|.$$

- *Coefficient of Determination (R^2).*

$$R^2 = 1 - \frac{\sum_{i=1}^{N \cdot T}(\mathbf{Y}_i - \widehat{\mathbf{Y}}_i)^2}{\sum_{i=1}^{N \cdot T}(\mathbf{Y}_i - \mathbf{Y})^2},$$

  where $\mathbf{Y}$ is the mean value of $\mathbf{Y}$, i.e., $\mathbf{Y} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{Y}_i$.
- *Explained Variance Score (Var).*

$$\mathrm{Var} = 1 - \frac{var\{\mathbf{Y} - \widehat{\mathbf{Y}}\}}{var\{\mathbf{Y}\}},$$

  where $var\{\cdot\}$ is the variance function.

5.1.2 Baselines

To evaluate the performance of our model, we compare it with following methods.

- *NAIVE*. We tested two NAIVE methods. One is the *historical average method*, which uses the average value of the last 12 time slices (3 hours) to predict the next value. The other is the *last-value method*, which uses the last value of the inputs to predict the next value. In our experiment, we find the last-value method performs better, so we report its result in the follows.
- *SVR* [44]. Linear Support Victor Regression uses a linear support vector machine for the regression task. We apply an open-source version[1].
- *ARIMA* [45]. Auto-Regressive Integrated Moving Average model is commonly used for modeling time series and has been widely adopted in time series prediction tasks. We use an ARIMA model with Kalman filter[2].
- *FC-LSTM* [46]. Long-short term memory network (LSTM) [26] is an improved RNN model that uses a "gate" mechanism to maintain long-term information. FC-LSTM is a special LSTM model with fully connected hidden units. We use the open source version[2].
- *DCRNN* [47]. Diffusion convolution recurrent neural network (DCRNN) uses graph convolution network to model spatial dependency and applies recurrent neural network to capture the temporal dynamics in an encoder-decoder manner. The code is public available[2].
- *T-GCN* [29]. Temporal graph convolutional network (T-GCN) combines GCN [31] and GRU [32]. Specifically, GCN is used to capture the topological structure of the graph to obtain the spatial dependence, and GRU is applied to explore the dynamic change of node attribute to mine the temporal dependence. The code is public available[1].
- *GraphWaveNet* [30]. It is a CNN-based method, which uses graph convolution [31] with dilated casual convolution [20]. The stacked dilated casual convolutions are used to capture temporal dependencies. The code is public available[3].
- *GNN-D* [14]. Graph neural network with decomposed cellular traffic (GNN-D) learns a node's representation by aggregating its neighbors through fully-connected neural networks and use RNN to predict future traffic.

5.1.3 Training Settings

Our model is implemented using Tensorflow 2.0 [48]. We apply Adam optimizer [49] with the learning rate of 0.001 on the mean square loss to train our model. To balance efficiency and effectiveness, in the spatial dependency modeling process, we set the aggregate depth $K = 3$ and the feature size of nodes is 12, i.e., the length r of traffic sequences. The output embedding size of the aggregator function r_a is 32. The output embedding sizes of the three layers r_k while $k = 1, 2, 3$ are 48, 24, and 12, respectively. For the temporal dynamic prediction process, the size of the convolution filters is set as 3, and the number of the convolution filters of each dilation convolution layer is set as 32. To cover the input sequence length, we use four residual blocks with a sequence of dilation factors $[1, 2]$ for each block. We set the input length as 12 in most of our experiments (we explain the reason in Section 5.4). To prevent our models from over-fitting, we adopt the dropout with $dropout\_rate = 0.1$ and $L2$ regularization with $\lambda = 0.0015$. Dropout is applied to the outputs of both the graph convolution layer and the temporal prediction layer. An early stopping strategy with a patience of 50 epochs on validation set is applied in our experiments. The dataset is split in chronological order for training, validation, and testing with the ratio of $0.6 : 0.1 : 0.3$. All comparative baselines are trained with the parameters following the description in the original papers or open-source codes.

5.2 Predictive Performance (RQ1)

In Table 2, we show the overall performance of our model, temporal models (SVR, ARIMA, and FC-LSTM), and spatio-temporal models (DCRNN, T-GCN, Graph WaveNet, GNN-D). We compare the performance of GTCN and baseline models for predicting traffic in last 15 minutes, 30

1. https://github.com/lehaifeng/T-GCN
2. https://github.com/liyaguang/DCRNN
3. https://github.com/nnzhan/Graph-WaveNet

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.

TABLE 2
Overall Prediction Performance of Our Model GTCN in Comparison With Baselines on Two Datasets MDT-Sub and MDT-Urb

Data	Models	Prediction Horizons															
		15 minutes				30 minutes				45 minutes				60 minutes			
		$rmse$	mae	R^2	Var	$rmse$	mae	R^2	Var	$rmse$	mae	R^2	Var	$rmse$	mae	R^2	Var
MDT-sub	NAIVE	0.5844	**0.1328**	0.7949	0.7949	0.6925	0.2234	0.7102	0.7102	0.7783	0.2630	0.6161	0.6161	0.8315	0.2827	0.5783	0.5783
	SVR	0.6021	0.2012	0.7027	0.7030	0.7133	0.2625	0.6545	0.6552	0.7769	0.2859	0.6226	0.6230	0.8246	0.3080	0.5805	0.5816
	ARIMA	0.5871	0.1964	0.7389	0.7394	0.6880	0.2407	0.6728	0.6736	0.7552	0.2721	0.6487	0.6495	0.8063	0.2934	0.6149	0.6155
	FC-LSTM	0.5744	0.1988	0.7625	0.7629	0.6532	0.2384	0.7094	0.7099	0.7074	0.2661	0.6754	0.6759	0.7415	0.2805	0.6416	0.6421
	DCRNN	0.5620	0.1975	0.7756	0.7756	0.6334	0.2286	0.7229	0.7229	0.6785	0.2437	0.7052	0.7053	0.7149	0.2690	0.6770	0.6772
	T-GCN	0.5573	0.1846	0.8021	0.8022	0.6159	0.2178	0.7526	0.7526	0.6500	0.2301	0.7216	0.7216	0.6966	0.2594	0.6971	0.6972
	GraphWaveNet	0.5514	0.1782	0.8045	0.8045	0.6044	0.2051	0.7826	0.7826	0.6483	0.2282	0.7523	0.7524	0.6885	0.2528	0.7207	0.7208
	GNN-D	0.5673	0.1798	0.7827	0.7830	0.6226	0.2108	0.7582	0.7585	0.6854	0.2352	0.7125	0.7129	0.7386	0.2609	0.6919	0.6925
	GTCN	**0.5311**	0.1689	**0.8306**	**0.8308**	**0.5853**	**0.1960**	**0.7942**	**0.7942**	**0.6259**	**0.2157**	**0.7646**	**0.7646**	**0.6632**	**0.2362**	**0.7356**	**0.7357**
MDT-urb	SVR	0.9655	0.5148	0.5874	0.5886	1.0413	0.5694	0.5323	0.5340	1.0844	0.6176	0.5033	0.5046	1.1113	0.6577	0.4902	0.4921
	ARIMA	0.9309	0.5091	0.6136	0.6145	1.0149	0.5505	0.5659	0.5669	1.0555	0.6074	0.5172	0.5185	1.0856	0.6433	0.5018	0.5032
	FC-LSTM	0.8882	0.4807	0.6578	0.6586	0.9698	0.5309	0.6032	0.6042	1.0129	0.5887	0.5540	0.5554	1.0467	0.6179	0.5466	0.5479
	DCRNN	0.8715	0.4704	0.6701	0.6702	0.9495	0.5235	0.6321	0.6321	0.9913	0.5693	0.5986	0.5987	1.0292	0.5993	0.5720	0.5720
	GraphWaveNet	0.8699	0.4592	0.6914	0.6914	0.9255	0.5141	0.6582	0.6582	0.9768	0.5507	0.6056	0.6056	1.0214	0.5893	0.5843	0.5843
	GNN-D	0.8771	0.4723	0.6806	0.6812	0.9491	0.5252	0.6445	0.6452	0.9933	0.5713	0.5921	0.5931	1.0302	0.5997	0.5891	0.5903
	GTCN	**0.8503**	**0.4479**	**0.7102**	**0.7102**	**0.9179**	**0.5008**	**0.6622**	**0.6623**	**0.9644**	**0.5525**	**0.6271**	**0.6272**	**0.9991**	**0.5763**	**0.5996**	**0.5997**

minutes, 45 minutes, and 60 minutes on MDT-sub and MDT-urb datasets. We list four metrics of all comparative methods. The Root Mean Squared Error (RMSE) and the Mean Absolute Error (MAE) are *log*-normalized. The T-GCN model's open-source code fails to deal with dataset MDT-urb due to a memory error because it is designed for a little graph with only 207 nodes [29], so we omit its result.

Our GTCN model obtains superior results on both datasets and outperforms other comparative models in most metrics for four prediction horizons, i.e., predicting the traffic trend in future 15 minutes, 30 minutes, 45 minutes, and 60 minutes, respectively. GTCN surpasses recent graph neural network models for spatio-temporal data, including DCRNN, T-GCN, and GraphWaveNet. The improvement of GTCN comparing with the second best performance model (GraphWaveNet) is around 3.2 to 6.6 percent. The RMSE reduction of GTCN comparing with Graph WaveNet is 3.7, 3.2, 3.5, and 3.7 percent for four prediction horizons, respectively. The MAE reduction of GTCN comparing with GraphWaveNet is 5.2, 4.4, 5.5, and 6.6 percent for four prediction horizons, respectively. Both GTCN and Graph WaveNet perform well on the 60-minute prediction horizon. We will discuss this in the results of long-term prediction later. GTCN significantly excels GNN-D, which is proposed to predict traffic on cell towers. The RMSE reduction of GTCN comparing with GNN-D is 6.4, 6.0, 8.7, and 10.2 percent for four prediction horizons, respectively. The MAE reduction of GTCN comparing with GNN-D is 6.1, 7.0, 8.3, and 9.5 percent for four prediction horizons, respectively. All these models perform better than temporal models including SVR, ARIMA, and FC-LSTM.

Results of Long-Term Prediction. Our model outperforms comparative models by a large margin on the 60-minute prediction horizon (predicting 4 future data points simultaneously). The improvement of GTCN increases when predicting the traffic in longer ranges. That indicates that our model captures long-term dynamics better than other models. It is due to the fact that GTCN can feed longer input sequences through stacked convolution layers to capture more information. GraphWaveNet also incorporates temporal convolutional layers, which might explain why GraphWaveNet performs better than other comparative models on the 60-minute prediction horizon.

Results of Next-Slice Prediction. For the 15-minute prediction horizon, the difference between different methods is slight. The reason is that the mobile data traffic varies slightly within the next time slice. Some simple models (like SVR, ARIMA) can also get high accuracy. However, once we extend the prediction horizon, the performance of most models decline fast, as shown in Table 2. We find that the NAIVE method, which regards the last-value of the inputs as the prediction result, performs better than most models on predicting traffic in next 15-minute horizon (the next time slice), but the performance degrades fast on prediction tasks for larger horizons. The reason is that the mobile traffic on each AP is really sparse, i.e., sometimes the traffic on an AP is zero. In such a situation, the NAIVE method outputs zero and it just fits the ground truth. But the prediction of the NAIVE method is useless for most applications, such

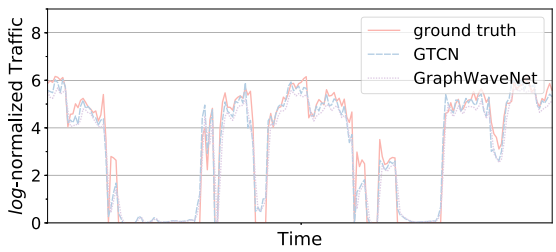


Fig. 6. Prediction versus the ground truth for a sample AP.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.

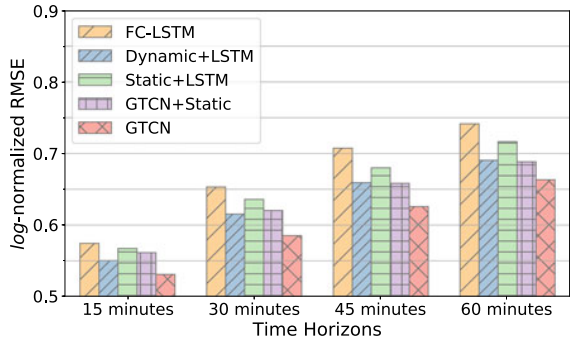


Fig. 7. Prediction errors with/without using spatial correlations.

as advance starting up and abnormal detection. We will show some results in Section 5.6.

We plot 15-minute predicted values versus real values of GTCN and the second best model GraphWaveNet on a snapshot of the testing dataset in Fig. 6. It shows that our GTCN generates more stable and low latency predictions than GraphWaveNet.

5.3 Benefits of Spatial Relation (RQ2)

The results in Table 2 have shown that spatio-temporal models (GTCN, DCRNN, T-GCN, Graph WaveNet, and GNN-D) outperform temporal models (SVR, ARIMA, and FC-LSTM) by a large margin. We further demonstrate the effective contribution of *dynamically* upgrading the feature of each node in the graph. Here we propose three comparing methods.

- *Dynamic+LSTM*. We connect our Node Information Aggregation module with an LSTM network, i.e., incorporating graph convolution into the FC-LSTM model. The setting of the LSTM network is the same as the FC-LSTM model.
- *Static+LSTM*. We use a static graph instead of the evolving User Mobility Graph in the Dynamic +LSTM method. The static graph considers uses' movements in the first month.
- *GTCN+Static*. We use a static graph instead of the evolving User Mobility Graph in our GTCN model.

The experiment is conducted on the dataset MDT-sub. Fig. 7 shows the performance of these models for four prediction horizons using the RMSE metric. Through modeling the spatial correlations of APs, our model GTCN has achieved a significant improvement in short-term, mid-term, and long-term forecasting. This indicates that our GTCN can effectively utilize spatial correlations to provide more accurate predictions. It also shows that our model is more capable of detecting spatial dependencies at each temporal stage than LSTM networks. The Dynamic+LSTM method, contrast to FC-LSTM and Static+LSTM, also performs better by incorporating the node information aggregation module. GTCN and Dynamic+LSTM, which use the evolving graph, perform better than GTCN+Static and Static+LSTM, respectively. The improvements in four prediction horizons are ranging from 6.2 to 13.9 percent. These results indicate that the evolving user mobility graph is more effective in modeling spatial relations between APs. The reason is that dynamically upgrading nodes' features benefits to concentrating on mining recent movement patterns of mobile users, and forgets redundant long-term memories.

5.4 Benefits of Involving Temporal Convolution (RQ3)

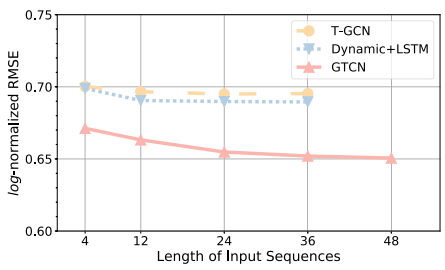
To evaluate the benefits of incorporating temporal convolutional layers into time series prediction tasks, we conduct experiments from two aspects: effectiveness and efficiency.

5.4.1 Long-term Receptive Field

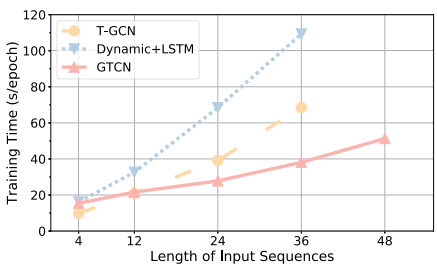
We compare our GTCN with Dynamic+LSTM (considered as using LSTM instead of temporal convolutional layers) and T-GCN (an RNN-based comparative model). We use traffic sequences with different lengths, from 4 (i.e., 60 minutes) to 48 (i.e., 12 hours), as inputs to forecast traffic in the future 60 minutes. The experiment is conducted on the dataset MDT-sub. Fig. 8a shows the performance of these models using the RMSE metric. The difference between these models is slight when the input sequence is short. When we use a longer sequence as input, the RMSE of our GTCN decreases faster than other models. It indicates that our GTCN can effectively capture information in long sequences by utilizing temporal convolutional layers. The improvement is 4.6 percent when the input length is 12, and 6.8 percent when the input length is 36. Considering efficient and fair comparison to all models, we set the input length as 12 in most of our experiments.

5.4.2 Model Efficiency

To see the benefits of the temporal convolution along the time axis in our proposal, we summarize the comparison of training time between GTCN, T-GCN, and Dynamic+LSTM with respect to different input lengths in Fig. 8b. We see RNN-based models need more training time as the length



(a) Prediction results w.r.t. different input length.



(b) Training time w.r.t. different input length.

Fig. 8. Benefits of involving temporal convolution.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.

TABLE 3
Computational Time When the Input Length is 36

Data	Models	Computational Time	
		Training (s/epoch)	Inference (s)
MDT-sub	T-GCN	68.49	17.45
	Dynamic+LSTM	109.26	18.53
	GTCN	38.12	2.08

of the input sequences increases. Then we summarize the comparison of training time and inference time when the input length is set as 36, as shown in Table 3. Our model achieves 8.4× improvement on inference time, which very benefits for low latency applications in practice. This acceleration of training and inference speed mainly benefits from applying the temporal convolution instead of recurrent structures, which can achieve parallel computation rather than relying on chain structures as RNN models do [21]. When the length of inputs increases to 48, T-GCN and Dynamic+LSTM have to decrease the batch size, since their GPU memory consumption exceeds the capacity, which will increase training batch and training time. So we omit the result under this circumstance.

5.5 Data Measurements and Impacts (RQ4)

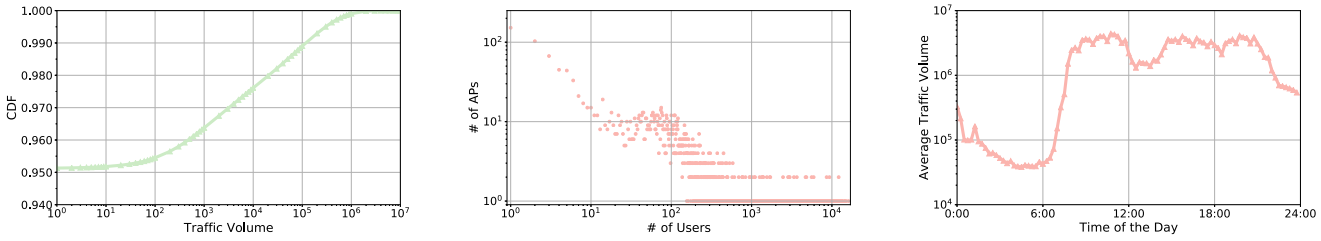
5.5.1 Impact of Mobile Data Traffic Volume

Fig. 9a plots the cumulative distribution function (CDF) of per one-quarter traffic volume in our MDT(sub) dataset. As shown, both light traffic ($<10^2$ bytes) and heavy traffic ($>10^5$ bytes) take up only a small percentage. We test the prediction ability of our model for different traffic volumes. Generally, it is more important to forecast heavy traffic than light traffic, because heavy traffic more impacts the stability and service quality of the whole wireless network. To evaluate the prediction performance for different traffic volume, we divide the testing set into five subsets based on traffic volume levels: $[0, 10^2)$, $[10^2, 10^3)$, $[10^3, 10^4)$, $[10^4, 10^5)$, $[10^5, \infty]$. Fig. 9d present the prediction performance for different traffic volume levels. For all prediction methods, *log*-normalized RMSE of light ($[0, 10^2)$) level is lowest, and *log*-normalized RMSE of $[10^2, 10^3)$, $[10^3, 10^4)$, and $[10^4, 10^5)$ levels are higher than *log*-normalized RMSE of heavy ($[10^5, \infty]$) level. The reason lies in the imbalance nature of the MDT dataset in which 95 percent of the traffic volume data is 0. The prediction results of all comparative methods have a tendency to close to 0. On the other hand, when the ground truth is in heavy level, the results will be close to the maximum of the dataset. Therefore, both light traffic and heavy traffic can be well predicted. Overall, the results indicate that our method outperforms others and it is applicable to predict mobile data traffic volume spanning a wide range. The improvements for different traffic volumes are ranging from -9.2 to 27.9 percent.

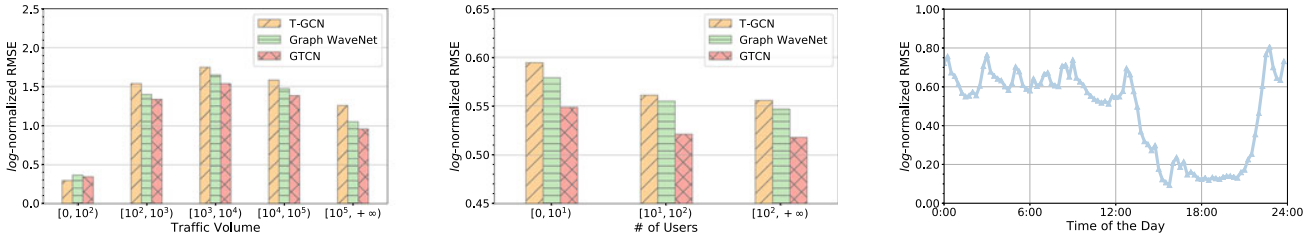
Maximal Capacity. In practice, each AP has a limited maximal capacity. Once the maximal capacity is reached, users will not be able to connect to this node so they have to connect to a far one. In this case, our model will predict the future traffic of these nodes. The results show that our model addresses this problem better than the comparative methods.

5.5.2 Impact of User Activity of APs

This experiment demonstrates the prediction performance of our method for APs with different levels of user activity. The user activity of an AP is used to measure how many users have connected to the AP. After checking the campus maps, we find that all of high activity APs are located in busy and crowded locations. Fig. 9b plots the number of APs with different user activity. Each point with coordinate (x, y) in the plot represents the number y of APs where x users have connected to. We divide the APs into three subsets according to user activity: low activity ($[0, 10^1)$), middle activity ($[10^1, 10^2)$), and high activity ($[10^2, \infty]$). We find all comparative methods achieve slightly better performance



(a) The CDF of one-quarter traffic volume. (b) The number of APs with different user activity. (c) The average traffic distribution of all APs within one day.



(d) The RMSE w.r.t. different traffic volume. (e) The RMSE w.r.t. different user activity. (f) The RMSE w.r.t. the time of a day.

Fig. 9. Impact of mobile data traffic volume, user activity, and time period.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.

TABLE 4
The Precision, Recall, and F1-Score of Detecting User Arrival and Leaving

Tasks	Metrics		
	precision	recall	F1-score
User Arrival	0.9530	0.9153	0.9338
User Leaving	0.9891	0.4889	0.6544

for higher activity APs than lower ones. The reason is that APs with high activity usually represent regular, daily, and periodical behaviors of users. They also show a high spatial dependency in their mobile data traffic on their neighboring APs. Therefore, such traffic patterns are easy to be captured. APs with low activity usually represent personal and occasional behaviors, which are not easy to be captured. Our method can capture the spatial dependency between these APs and their neighbors to improve the prediction performance. Therefore, it achieves a lower prediction error than other methods. Overall, the results indicate that our method achieves better performance than comparative methods over all ranges of user mobility of APs. The improvements for different user mobility are ranging from 6.7 to 9.4 percent.

5.5.3 Impact of Time Period

This experiment shows the temporal trend of the prediction errors of the selected methods. Since both the traffic volume and users' traffic usage preference vary with time, we try to demonstrate how these factors impact prediction performance. Fig. 9c plots the average traffic volume for a whole day. Most users use wireless networks from around 8:00 to 22:00. Fig. 9f plots the *log*-normalized RMSE metric of the prediction performance of our method during one day. According to the result, we demonstrate that our model has a better prediction ability for daytime mobile data traffic. The reason lies in the daily and periodical behaviors of users during the daytime. Users' personal behaviors during the night are hard to predict. We find the predict performance gets better from around 9:00. After checking the school timetable, we find that many courses start after 9:00 and many users begin to move to high mobility APs, which are at crowded locations and have large mobile data traffic, at that time. As we mentioned before, such daily patterns are easier to capture. Therefore, our method can get better prediction performance and achieves smaller RMSEs.

5.6 Application for Energy-Saving (RQ5)

Commercial WiFi networks need to remain active for better service quality. However, a commercial WiFi network may contain thousands of active APs all day, which waste lot of energy. If we can accurately predict when users arrive and leave and turn the AP into different modes (normal mode or low-power mode) ahead, it will be useful for energy saving while keeping satisfied service quality. To achieve this target, we demonstrate the ability for predicting users' arrival and leaving, as shown in Table 4. Our method achieves precision with 0.9530 and recall with 0.9153 on predicting user arrival, and precision with 0.9891 and recall with 0.4889 on predicting user leaving. It means less than 1 in 10 arrival prediction is a false negative, and around 1 in 100 leaving prediction is a false positive. It will still keep high service quality if we turn the APs into different modes according to our predicted results.

We compute the amount of energy saved by such a strategy. Assuming low-power mode does not cost any energy, for each AP, if busy time accounts for $x\%$ proportion, we can save as almost $1 - x\%$ energy. The recall of predicting user leaving is 0.4889 (see Table 4), which means around 48.89 percent relax time is accurately predicted by our method. Therefore, we can roughly estimate that $0.4889(1 - x\%)$ energy can be saved by turning APs' mode ahead.

6 Conclusion

In this paper, we proposed a deep neural network model GTCN to predict mobile data traffic of APs in large-scale wireless networks. GTCN first uses a graph convolutional neural network to model the spatial correlations between APs by exploring spatio-temporal mobility trajectories of mobile users. Then, we adapted temporal convolutional network layers to model the temporal trend of mobile data traffic on each AP. Experimental results on two real-world datasets demonstrate the efficiency and efficacy of our GTCN. The datasets will be released soon.

Acknowledgments

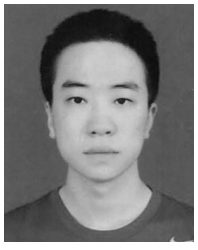
This work was supported in part by Shenzhen Basic Research Grant (JCYJ20170816100819428), National Key R&D Program of China (2018YFC0830500), National Natural Science Foundation of China (61922067, U1736205, 61902305), MoE-CMCC "Artifical Intelligence" Project (MCM20190701), National Science Basic Research Plan in Shaanxi Province of China (2019JM-159), National Science Basic Research Plan in Zhejiang Province of China (LGG18F020016). The work of John C.S. Lui was supported in part by RIF R4032-18.

References

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2017–2022 White Paper," 2019.
[2] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, Third Quarter 2019.
[3] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," 2018, *arXiv:1809.08707*.
[4] "ZTE, china unicom build ai-powered low carbon mobile network," 2020. [Online]. Available: https://disruptive.asia/zte-china-unicom-ai-powered-low-carbon-mobile-network/
[5] F. Yang, Y. Jiang, T. Pan, and X. E., "Traffic anomaly detection and prediction based on SDN-enabled ICN," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2018, pp. 1–5.
[6] K. Sugiyama, J. Urakawa, M. Taya, A. Yamada, A. Kobayashi, and A. Tagami, "Empirical analysis of customer behavior for tiered data plans in mobile market," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2016, pp. 389–394.
[7] Y. Shu, M. Yu, J. Liu, and O. W. Yang, "Wireless traffic modeling and prediction using seasonal arima models," in *Proc. IEEE Int. Conf. Commun.*, 2003, pp. 1675–1679.
[8] W.-C. Hong, "Application of seasonal SVR with chaotic immune algorithm in traffic flow forecasting," *Neural Comput. Appl.*, vol. 21, no. 3, pp. 583–593, 2012.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.

[9] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang, "Geospatial and temporal dynamics of application usage in cellular data networks," *IEEE Trans. Mobile Comput.*, vol. 14, no. 7, pp. 1369–1381, Jul. 2015.
[10] J. Wang *et al.*, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.
[11] J. Feng, X. Chen, R. Gao, M. Zeng, and Y. Li, "DeepTP: An end-to-end neural network for mobile cellular traffic prediction," *IEEE Netw.*, vol. 32, no. 6, pp. 108–115, Nov./Dec. 2018.
[12] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *Proc. 19th Int. Workshop Mobile Comput. Syst. Appl.*, 2018, pp. 231–240.
[13] L. Fang, X. Cheng, H. Wang, and L. Yang, "Mobile demand forecasting via deep graph-sequence spatiotemporal modeling in cellular networks," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3091–3101, Nov./Dec. 2018.
[14] X. Wang *et al.*, "Spatio-temporal analysis and prediction of cellular traffic in metropolis," *IEEE Trans. Mobile Comput.*, vol. 18, pp. 2190–2202, Sep. 2018.
[15] C. Zhang, H. Zhang, D. Yuan, and M. Zhang, "Citywide cellular traffic prediction based on densely connected convolutional neural networks," *IEEE Commun. Lett.*, vol. 22, no. 8, pp. 1656–1659, Aug. 2018.
[16] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.
[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[18] S. Wu, Y. Zhang, C. Gao, K. Bian, and B. Cui, "GARG: Anonymous recommendation of point-of-interest in mobile networks by graph convolution network," *Data Sci. Eng.*, vol. 5, no. 4, pp. 433–447, 2020.
[19] F. Xu, Y. Li, M. Chen, and S. Chen, "Mobile cellular big data: linking cyberspace and the physical world with social ecology," *IEEE Netw.*, vol. 30, no. 3, pp. 6–12, May/Jun. 2016.
[20] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proc. Int. Conf. Learn. Representations*, 2016.
[21] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*.
[22] R. Li, Z. Zhao, X. Zhou, J. Palicot, and H. Zhang, "The prediction analysis of cellular radio access network traffic: From entropy theory to networking practice," *IEEE Commun. Mag.*, vol. 52, no. 6, pp. 234–240, Jun. 2014.
[23] Y. Xu, W. Xu, F. Yin, J. Lin, and S. Cui, "High-accuracy wireless traffic prediction: A gp-based machine learning approach," in *Proc. IEEE Global Commun. Conf.*, 2017, pp. 1–6.
[24] R. Li, Z. Zhao, J. Zheng, C. Mei, Y. Cai, and H. Zhang, "The learning and prediction of application-level traffic data in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3899–3912, Jun. 2017.
[25] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.
[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[27] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3634–3640.
[28] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 922–929.
[29] L. Zhao *et al.*, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3848–3858, Sep. 2020.
[30] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 1907–1913.
[31] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.
[32] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.
[33] Z. Diao, X. Wang, D. Zhang, Y. Liu, K. Xie, and S. He, "Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 890–897.
[34] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1655–1661.
[35] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang, "Urban traffic prediction from spatio-temporal data using deep meta learning," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 1720–1730.
[36] T. Li, J. Zhang, K. Bao, Y. Liang, Y. Li, and Y. Zheng, "Autost: Efficient neural architecture search for spatio-temporal prediction," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2020, pp. 794–802.
[37] P. Wang, F. Sun, D. Wang, J. Tao, X. Guan, and A. Bifet, "Inferring demographics and social networks of mobile device users on campus from ap-trajectories," in *Proc. 26th Int. Conf. World Wide Web Companion*, 2017, pp. 139–147.
[38] P. Wang, F. Sun, D. Wang, J. Tao, X. Guan, and A. Bifet, "Predicting attributes and friends of mobile users from ap-trajectories," *Inf. Sci.*, vol. 463, pp. 110–128, 2018.
[39] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Trans. Nucl. Sci.*, vol. 44, no. 3, pp. 1464–1468, Jun. 1997.
[40] M. J. Dworkin, "SHA-3 standard: Permutation-based hash and extendable-output functions," 2015.
[41] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Advances Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.
[42] A. V. D. Oord *et al.*, "Wavenet: A generative model for raw audio," 2016, *arXiv:1609.03499*.
[43] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.
[44] H. Drucker, C. J. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. Advances Neural Inf. Process. Syst.*, 1997, pp. 155–161.
[45] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results," *J. Transp. Eng.*, vol. 129, no. 6, pp. 664–672, 2003.
[46] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[47] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. Int. Conf. Learn. Representations*, 2018.
[48] "Tensorflow 2.0," [Online]. Available: https://www.tensorflow.org/api_docs/python/tf
[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.



Feiyang Sun received the BS degree in automation engineering from Xi'an Jiaotong University, Xi'an, China, in 2015. He is currently working toward the graduate degree at the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University. His research interests include Spatio-temporal data modeling, location-based social network, time series modeling, and user profiling.

Pinghui Wang received the BS and PhD degrees in information engineering from Xi'an Jiaotong University, Xi'an, China, in 2006 and 2012, respectively. He is currently a professor with the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University, and is also with Shenzhen Research Institute of Xi'an Jiaotong University, Shenzhen, China. His research interests include Internet traffic measurement and modeling, traffic classification, abnormal detection, and online social network measurement.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.

Junzhou Zhao received the BS and PhD degrees in control science and engineering from Xi'an Jiaotong University, in 2008 and 2005, respectively. He is currently an associated professor at the School of Cyber Science and Engineering, Xi'an Jiaotong University. His research interests include graph data mining and streaming data processing.

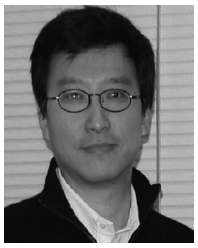
Nuo Xu received the BS degree in automation engineering from Xi'an Jiaotong University, Xi'an, China, in 2017. He is currently working toward the graduate degree at the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University. His research interests include natural language processing (NLP), graph neural network, and bioinformatics.

Juxiang Zeng received the BS degree in automation engineering from Xi'an Jiaotong University, Xi'an, China, in 2017. She is currently working toward the graduate degree at the MOE Key Laboratory for Intelligent Networks and Network Security, Xi'an Jiaotong University. Her research interests include computer vision, artificial intelligence, and graph mining.

Jing Tao received the BS and MS degrees in automatic control from Xi'an Jiaotong University, Xi'an, China, in 2001 and 2006, respectively. He is currently a teacher in Xi'an Jiaotong University and on-the-job PhD degree at the Systems Engineering Institute. His research interests include Internet traffic measurement and modeling, traffic classification, abnormal detection, and botnet.

Kaikai Song received the BS and PhD degrees from the University of Science and Technology of China, Hefei, China, in 2013 and 2018, respectively. He is currently a researcher at the Huawei Noah's Ark lab. His research interests include computer vision and artificial intelligence for ICT operations.

Chao Deng received the MS and PhD degrees from the Harbin Institute of Technology, Harbin, China, in 2003 and 2009, respectively. He is currently a deputy general manager at the AI Center of China Mobile Research Institute. His research interests include artificial intelligence for ICT operations.



John C.S. Lui (Fellow, IEEE) received the PhD degree in computer science from UCLA. He is currently a professor at the Department of Computer Science and Engineering, Chinese University of Hong Kong. His current research interests include communication networks, network system security, network economics, network sciences, cloud computing, large-scale distributed systems, and performance evaluation theory.

Xiaohong Guan (Fellow, IEEE) received the BS and MS degrees in automatic control from Tsinghua University, in 1982 and 1985, respectively, and the PhD degree in electrical engineering from the University of Connecticut, in 1983. He is currently a professor at the Systems Engineering Institute, Xi'an Jiaotong University, Xi'an, China. He was appointed Cheung Kong Professor of Systems Engineering in 1999, and dean of the School of Electronic and Information Engineering in 2008. He has been the director of the Center for Intelligent and Networked Systems since 2001, Tsinghua University.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on July 08,2023 at 09:31:30 UTC from IEEE Xplore. Restrictions apply.