Real-Time Traffic Speed Estimation With Graph Convolutional Generative Autoencoder

James Jian Qiao Yu, *Member, IEEE*, and Jiatao Gu

***Abstract*— Real-time traffic speed estimation is an essential component of intelligent transportation system (ITS) technologies. It is the foundation of modern transportation control and management applications. However, the existing traffic speed acquisition systems can only provide real-time speed measurements of a small number of roads with stationary speed sensors and crowdsourcing vehicles. How to utilize this information to provide traffic speed maps for transportation networks is becoming a key problem in ITSs. In this paper, we present a novel deep-learning model called graph convolutional generative autoencoder to fully address the real-time traffic speed estimation problem. The proposed model incorporates the recent development in deep-learning techniques to extract the spatial correlation of the transportation network from the input incomplete historical data. To evaluate the proposed speed estimation technique, we conduct comprehensive case studies on a real-world transportation network and vehicular traces. The simulation results demonstrate that the proposed technique can notably outperform existing traffic speed estimation and deep-learning techniques. In addition, the impact of dataset properties and control parameters is investigated.**

***Index Terms*— Traffic estimation, deep learning, generative adversarial network, graph convolutional network, data-driven model.**

I. Introduction

INTELLIGENT transportation system (ITS) is among the most important components in future smart cities. Contributed by the advanced sensing, communication, and computation techniques, ITS is expected to greatly improve the efficiency of general traffic [1]–[3]. In recent years, the transportation industry and research community have witnessed a plethora of effort on how to optimally control the traffic and operate vehicles to achieve various social objectives, e.g., reduce greenhouse gas emission [4]–[6] and traffic congestion [7], [8]. In order to have a better knowledge and control on the traffic, these ITS applications rely heavily on the real-time traffic speed of each road in transportation networks [9]–[12]. For instance, real-time route planning service employs the traffic speed to guide vehicles to avoid congested roads, which in turn reduces both the driving time and the pollutant emission [3].

Manuscript received May 26, 2018; revised December 16, 2018 and April 2, 2019; accepted April 8, 2019. The Associate Editor for this paper was Y. Lv. *(Corresponding author: James Jian Qiao Yu.)*

J. J. Q. Yu is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: yujq3@sustech.edu.cn).

J. Gu is with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong.

Digital Object Identifier 10.1109/TITS.2019.2910560

Existing industrial solutions to provide real-time traffic speed estimations of the transportation network mainly involve stationary speed sensors and vehicular global positioning system (GPS) records. Take Google Maps as an example. The service provider gathers massive real-time vehicular GPS records containing the position and driving speed of crowdsourcing vehicles [13]. These data are then aggregated to construct a traffic speed map, which is shown in Google Maps and is employed to provide real-time routing service. Nonetheless, this model requires a huge volume of information from crowdsourcing drivers, which is unavailable for most other services. Furthermore, since the records are typically labeled by driver identifiers, user information privacy may be infringed. It is also highly possible that roads in rural areas cannot be consistently covered by crowdsourcing vehicles, rendering incomplete speed data.

Due to the importance of real-time traffic speed in transportation system applications, much research effort has been made to devise algorithms for providing real-time traffic speed with limited measurements, which is practical as a general-purpose solution. A wide variety of methods have been employed to address this real-time traffic speed estimation problem (sometimes called traffic data imputation problem [9]). These methods can be classified into at least two main categories, namely, black-box statistical methods (e.g., [9], [14], [15]) and traffic-model-based methods (e.g., [16]–[18]). Both class of methods have their own merits, and transportation system applications require the right methods. For instance, statistical approaches are generally more robust to noise and perturbations, and are thus better for traffic estimation and prediction in which under-determinacy is not a significant problem. Meanwhile, model-based approaches can be better adopted in what-if reasoning, decision impact assessments and network dynamics research, etc. A wide variety of methods have been employed to address this real-time traffic speed estimation problem (also called traffic data imputation problem [9]). Additionally, many results are published on developing predictions of traffic flow data, e.g., [19]. These methods can also help design traffic speed estimation approaches.

However, there exists a research gap in the real-time traffic speed estimation problem, especially when developing the speed for each road in a large area, i.e., traffic speed map. Most existing methods estimate the speed for an arbitrary road solely by learning the complete or partial historical measurements, see [9], [14] for examples. Such methods do not utilize the spatial correlations among real-time traffic

speeds of adjacent roads, which can potentially help improve the estimation accuracy. In addition, these models typically can only address one or few roads at one time. This means that traffic speed maps can only be constructed by executing the estimation algorithms independently for each road, which can be highly computationally inefficient. Furthermore, existing work relies heavily on stationary speed sensor records on a same road, rendering mobile vehicular GPS records underutilized. Very recent work on traffic speed prediction (e.g., [20]) extracts the spatial correlation characteristics of transportation networks by adopting the convolution idea. Nonetheless, the proposed diffusion-based approach cannot be employed to handle vehicular GPS records, which lead to intermittent speed measurements for arbitrary roads. Thus the technique still suffers from the same data under-utilization issue in traffic speed estimation problem.

To bridge the research gap, in this work we propose a new deep neural network architecture called graph convolutional generative autoencoder (GCGA) to address the real-time speed estimation problem in modern cities. We specifically consider practical scenarios in which a small number of crowdsourcing vehicles provide their real-time GPS records for speed estimation over a large region. Different from existing work, the proposed model can extract the graph-related spatial characteristics of transportation networks to develop traffic speed maps at one time. In addition, this model can relax the dependency on stationary speed sensors and fully utilize the dynamic, independent, and incomplete vehicular GPS records. In this architecture, we adopt the graph convolution concept from graph convolutional network (GCN) [21] for feature extraction. In addition, we formulate the network as a variant of generative adversarial network (GAN) [22] to generate traffic speed maps. Contributed by the superior data generation capability of the generative model [23], i.e., GAN in this work, the proposed model can develop more accurate and robust speed maps compared with typical deep learning models. As far as we are concerned, this is the pioneer work on real-time traffic speed map generation with rare measurements.

The main contributions of this work are summarized below:

- We propose a novel GCGA as a general-purpose feature-generating methodology for graphs. We adopt the design principle of GCN and GAN to formulate the architecture of GCGA, and a practical training method is proposed.
- We employ GCGA to address the real-time traffic speed estimation problem. We analyze the properties of the problem and discuss the implementation details of the GCGA-based speed estimation approach.
- We develop a tailor-made mechanism to boost the training speed of the proposed system. The mechanism significantly reduces the training time while maintains the system performance.

The rest of this paper is organized as follows. In Section II, we first give the mathematical formulation of the investigated traffic speed estimation problem, and discuss the recent related literature. We formulate the proposed GCGA and discuss its application in addressing the traffic speed estimation problem in Section III. Section IV presents the results and discussions of the case studies, and this paper is concluded in Section V.

TABLE I
DEFINED SYMBOLS IN THE SYSTEM MODEL

Symbol	Definition
$\mathcal{G}$	Transportation network.
$\mathcal{N}$	Set of road intersections in $\mathcal{G}$.
$\mathcal{E}$	Set of roads in $\mathcal{G}$.
$\mathcal{T}$	Set of time slots in the past and present.
$\mathcal{S}$	Set of roads equipped with stationary speed sensors.
$\mathcal{P}$	Additional road network information.
$v_{e,t}$	Ground truth traffic speed of road e at time t.
$\mathrm{fr}(e)$	Starting node of road e.
$\mathrm{to}(e)$	Ending node of road e.
$\hat{v}_{e,t}$	Estimated traffic speed of road e at time t.
$\mathcal{C}_t$	Roads traversed by crowdsourcing vehicles at time t.
$\mathcal{E}_t^+$	Roads whose traffic speeds are observed by the system operator at time t.
$\mathcal{V}_t^+$	Observed speeds of roads in $\mathcal{E}_t^+$ at time t.
$\hat{\mathcal{V}}_t$	Estimated speeds of all roads in $\mathcal{E}$ at time t.

II. TRAFFIC SPEED ESTIMATION PROBLEM

In this section, we define the traffic speed estimation problem. The objective and input data are formalized with mathematical models, and related work is discussed.

A. System Model

The symbols defined in this section are summarized in Table I. We first model the investigated road network as a directed graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of road intersections (vertices in the graph), and $\mathcal{E}$ is the set of roads (edges). Given a transportation network, the corresponding graph $\mathcal{G}$ can be constructed in a two-step process. We first extract all road intersections in the network and create corresponding nodes in the graph. Then each road is included in the network by creating an edge from its starting node to its ending one. Let $\mathrm{fr}(e) \in \mathcal{N}$ and $\mathrm{to}(e) \in \mathcal{N}$ be the starting and ending nodes of a road $e \in \mathcal{E}$. Considering infinite discrete time slots in the past and present $\mathcal{T} = \{\cdots, -2, -1, 0\}$ where 0 stands for the current one, the average traffic flow speed of a road $e \in \mathcal{E}$ at time $t \in \mathcal{T}$ is denoted by $v_{e,t}$. Additionally, roads in the network have their own properties, e.g., speed limits and numbers of nearby point of interest, denoted by $\mathcal{P}$. Among all roads in the network, the system operator can only observe the traffic speed of some of them, which are denoted by $\mathcal{E}_t^+$. We further use set $\mathcal{V}_t^+$ to stand for the observed traffic speed of each $e \in \mathcal{E}_t^+$.

In this road network, the traffic speed of each road can be observed from two sources: crowdsourced vehicular GPS records and optional stationary speed sensors. Let $\mathcal{S}$ be the set of roads equipped with speed sensors, and $\mathcal{C}_t$ be the set of roads on which vehicles traverse and report their GPS records to the system operator at t. Consequently, we have $\mathcal{E}_t^+ = \mathcal{S} \cup \mathcal{C}_t, \forall t \in \mathcal{T}$, which may change for different t values with vehicles traversing the transportation network.

The objective of the traffic speed estimation problem is to employ a traffic speed estimator $\mathrm{E}(\cdots)$ to estimate the current traffic speed of all roads in the transportation network:

$$\hat{\mathcal{V}}_0 = \mathrm{E}(\mathcal{V}_0^+, \mathcal{V}_{-1}^+, \cdots, \mathcal{P}), \tag{1}$$

where $\hat{\mathcal{V}}_0 = \{\hat{v}_{e,0} | e \in \mathcal{E}\}$ is the set of traffic speed estimations, and $\hat{v}_{e,0}$ is the estimated current traffic speed for road e.

The quality of estimated speed values can be evaluated using the mean absolute percentage error (MAPE) of the unobserved roads:

$$\text{MAPE} = \frac{1}{|\mathcal{E}| - |\mathcal{E}_0^+|} \sum_{e \in \mathcal{E} \setminus \mathcal{E}_0^+} \frac{|\hat{v}_{e,0} - v_{e,0}|}{v_{e,0} + \xi}, \tag{2}$$

where ξ is a small positive value[1] which prevents the divide-by-zero problem if $v_{e,0} = 0$.

B. Related Work

Real-time traffic speed estimation and prediction is regarded as a key component in ITS. Many statistical and learning approaches have been employed to solve the problem. A widely adopted technique is the auto-regressive integrated moving average (ARIMA) approach, which captures the temporally related features from time-series data. For example, references [24], [25] are some representative work on employing ARIMA and its variants in providing real-time traffic speed with historical speed measurements. Given complete input data, such techniques can generate both the current speed estimations and future predictions. Using the time-series analysis idea, other time-series models are also proposed to address this problem, see [26], [27] for examples. These approaches require the complete historical speed measurements to generate speed estimations and predictions, rendering it difficult to develop a complete traffic speed map with limited speed sensor coverages.

Besides, nearest neighborhood interpolation approaches are also widely used in the literature to handle the incomplete measurement issue. Conventional solutions adopt k-NN methods to estimate traffic speeds of arbitrary roads with known speed measurements from its spatially adjacent roads, see [28], [29] for examples. Recent work incorporates other traffic related properties to perform matrix factorization methods [30], [31]. These approaches intrinsically assume that adjacent roads have similar speeds, which is typically non-trivial [32]. There are also results published on employing data from multiple heterogeneous sources for traffic speed estimation and prediction. Interested readers may refer to [33] for a detailed review.

In recent years, researchers have been focusing on employing machine learning and neural network techniques in traffic speed/flow estimation and prediction [19]. Reference [9] proposes an autoencoder-based deep neural network to estimate the missing speed values in the time-series traffic speed measurements, and the stacked variant of the network is capable of providing reliable future traffic flow predictions [19]. While conventional neural network-based approaches tend to let the network learns the topological characteristics of the transportation network from raw input data [34], [35], recent related work integrates the adjacency information as an input of the model, which can significantly release the feature extraction capabilities of the technique [20], [36], [37]. Nonetheless, these approaches are designed for the traffic speed prediction problem with full historical data, thus cannot be directly employed to give speed estimations with incomplete measurements.

[1]We set $\xi = 0.01$ km/h in the case studies.

TABLE II
DEFINED SYMBOLS IN GRAPH CONVOLUTIONAL GENERATIVE AUTOENCODER

Symbol	Definition
	Generative Adversarial Network
G,D	Generator and discriminator.
θ^{G},θ^{D}	Network parameters of G and D.
$\mathrm{G}(\cdot,\cdot)$,$\mathrm{D}(\cdot,\cdot)$	Mathematical representation of G and D.
$\mathbf{x}$,$\mathbf{z}$	Input and noise data.
$\mathbb{P}_{\mathbf{x}}$,$\mathbb{P}_{\mathbf{z}}$	Probability distribution of $\mathbf{x}$ and $\mathbf{z}$.
	Graph Convolutional Network
N,F	Number of input and output features.
X,Z	Input and output feature matrix.
L	Number of GCN layers.
$H^{(l)}$	Output matrix of the l-th layer.
$f(\cdot,\cdot)$	Propagation rule of GCN.
$\sigma(\cdot)$	Non-linear activation function.
$\hat{A}$	Adjacency matrix with ones at diagonal entries.
$\hat{D}$	Diagonal node degree matrix of $\hat{A}$.
$W^{(l)}$,$b^{(l)}$	Layer-specific weight and bias matrices of the l-th layer.
	Graph Convolutional Generative Autoencoder
C	Number of available input data matrices.
$X_{(c)}$	The c-th input feature matrix.
$\mathcal{R}^+_{(c)}$,$\mathcal{R}^-_{(c)}$	Available and missing feature nodes in $X_{(c)}$.
$\mathcal{X}$	Augmented input data matrices.
M	Number of new data matrices developed for each $X_{(c)}$.
$X^-_{(c),m}$	The m-th new input data matrix based on $X_{(c)}$.
$\tilde{\mathcal{R}}^-_{(c),m}$	Nodes whose features are removed from $X_{(c)}$ to generate $X^-_{(c),m}$.
$\tilde{\mathcal{R}}^+_{(c),m}$	Nodes whose features are retained in $X^-_{(c),m}$.
$\hat{X}^-_{(c),m}$	Estimate of $X_{(c)}$ by GCGA using $X^-_{(c),m}$.
L^{G}	The MSE loss function of the generator.
$L^{\mathrm{D}}_{\mathbf{G}\mathbf{z}}$,$L^{\mathrm{D}}_{\mathbf{x}}$	The BCE loss functions of the discriminator on all $X^-_{(c),m}$ and $X_{(c)}$.
$\mathcal{G}^*$	Transformed road-based transportation network.
$\mathcal{N}^*$	Set of roads in $\mathcal{G}$.
$\mathcal{E}^*$	Connectivity of roads in $\mathcal{G}$.
W, H	Sub-region width and height.

III. GRAPH CONVOLUTIONAL GENERATIVE AUTOENCODER

As discussed in the previous section, existing solutions to provide real-time estimated traffic speed suffer from drawbacks. In this section, we propose a new data imputation technique based on the recent development of deep learning techniques. We first give a brief introduction on GAN and GCN. Then we formulate the proposed GCGA model and discuss its training method. Finally, how to employ GCGA to handle the traffic speed estimation problem is elaborated. The symbols used in this section is summarized in Table II for reference.

A. Generative Adversarial Network and Graph Convolutional Network

1) Generative Adversarial Network: GAN [22] is a recent deep neural network framework aiming to generate artificial samples indistinguishable from their real counterparts. This objective is achieved by designing a generative and a discriminative neural network to formulate a two-player minimax game. In this game, the generative network, called *Generator*, generates random new samples while the other adversarial network, called *Discriminator*, evaluates these samples for authenticity.

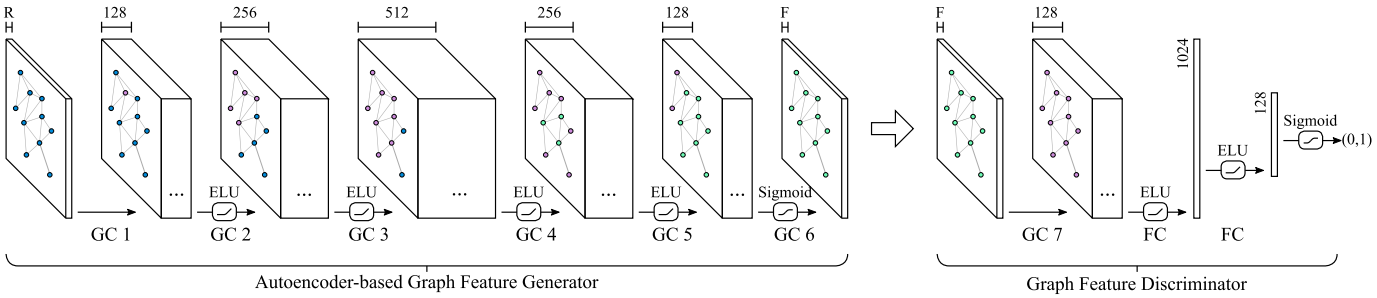


Fig. 1. Illustration of the proposed GCGA model architecture.

Let G and D be the generator and discriminator, respectively, which are parameterized by θ^{G} and θ^{D}. $\mathrm{G}(\cdot,\cdot)$ and $\mathrm{D}(\cdot,\cdot)$ are the mathematical representations of G and D, respectively. Given a group of real data $\mathbf{x}$ which follows a distribution $\mathbb{P}_{\mathbf{x}}$, GAN tries to establish a non-linear mapping between the real data and a group of noise data $\mathbf{z} \sim \mathbb{P}_{\mathbf{z}}$, where $\mathbb{P}_{\mathbf{z}}$ is trivial. The generator and discriminator play the following two-player minimax game with objective function $V(\mathrm{G},\mathrm{D})$ [22]:

$$\min_{\theta^{\mathrm{G}}}\max_{\theta^{\mathrm{D}}} V(\mathrm{G},\mathrm{D}) = \mathbb{E}_{\mathbb{P}_{\mathbf{x}}}[\log \mathrm{D}(\mathbf{x},\theta^{\mathrm{D}})] + \mathbb{E}_{\mathbb{P}_{\mathbf{z}}}[\log(1-\mathrm{D}(\mathrm{G}(\mathbf{z},\theta^{\mathrm{G}}),\theta^{\mathrm{D}}))]. \quad (3)$$

In the typical implementation of GAN, both G and D are constructed with neural networks [22], [38]. At the beginning of the GAN training process, G has randomized initial parameters θ^{G}, which make D reject $\mathrm{G}(\mathbf{z},\theta^{\mathrm{G}})$ easily. In such a case, both $V(\mathrm{G},\mathrm{D})$ and the second term of (3), i.e., the loss function of G, have large values. The training process tries to adjust θ^{G} to let G generate realistic samples to fool D, rendering a low loss value for G. In the meantime, θ^{D} is also adjusted to make D successfully distinguish the new and better samples from G. This procedure repeats until G is good enough to circumvent the rejection of D, and the real data distribution $\mathbb{P}_{\mathbf{x}}$ is learned by G in this process. Interested readers may refer to [38]–[40] for detailed introduction and theoretical analyses of GAN.

2) Graph Convolutional Network: GCN [21] refers to a neural network model which aims at extracting features from graphs. It inherits the idea of *convolution filter* from typical convolutional neural networks (CNN) aiming at image pixels or arrays of signals in Euclidean space. The filter performs neighborhood mixing on the source data with its uniform receptive field, leading to shared information in the result [23]. When the receptive field moves over the source data, the filter parameters remains constant. However, receptive field parameters cannot be easily shared in non-Euclidean structures such as graphs, since the node connectivity is irregular [41]. This limits the application of CNN to graphs.

GCN follows the idea of CNN and adopts the connectivity structure of the input graph as the convolution filter for neighborhood mixing [21], [42]. This model tries to learn the features on a graph $\mathcal{H}(\mathcal{R},\mathcal{A})$ which takes 1) an $|\mathcal{R}| \times N$ feature matrix X and 2) a representative description of the graph structure, e.g., adjacency matrix A as input data, and develop an $|\mathcal{R}| \times F$ node-level output feature matrix Z, where N and F are the number of input and output features for each node in the graph. In an L-layer GCN, each layer of the network can be expressed as a non-linear function:

$$H^{(l+1)} = f(H^{(l)}, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)} + b^{(l)}), \quad (4)$$

where $H^{(l)}$ is the output matrix of the l-th layer, $f(\cdot,\cdot)$ is the GCN propagation rule, $\sigma(\cdot)$ is the non-linear activation function, $\hat{A} = \{\hat{a}_{ij}\} = A + I$, $\hat{D} = \{\hat{d}_{ij}\}$ is the diagonal node degree matrix of $\hat{A}$, and $W^{(l)}$ and $b^{(l)}$ are a layer-specific tunable weight and bias matrices, respectively. Obviously, $H^0 = X$, $H^L = Z$, and $\hat{d}_{ii} = \sum_j \hat{a}_{ij}$. This propagation rule is actually motivated by the first-order approximation of Chebyshev polynomials of eigenvalues in the spectral domain on the input graph [21], [43], which makes the computation fast and easy. It has been demonstrated that the approximation can still lead to highly competitive results in graph-learning datasets [21].

B. Graph Convolutional Generative Autoencoder

While GAN and GCN have demonstrated their efficacy in handling data generation and graph feature extraction problems, they cannot individually overcome the main challenge of the traffic speed estimation problem, i.e., *how to generate realistic features (traffic speed) for each node/edge in a graph given the graph topology and partial feature information*. In this section, we propose a GCGA model to fully address this challenge by adopting the design principle of GAN and GCN.

The architecture of the proposed GCGA model is presented in Figure 1. GCGA is composed of two adversarial neural networks, namely, an autoencoder-based graph feature generator and a graph feature discriminator. Both networks participate in the minimax game defined by GAN to improve the authenticity of the generated graph features. After fine-tuning the network parameters via training, the generator provides complete and realistic node features as output based on the partial feature data input.

1) Feature Generation: At the beginning of the feature generation process, the available features in the input graph are normalized to $[0, 1]$, and the missing feature values are assigned with zeros. The resulting zero-padded $|\mathcal{R}| \times N$ graph feature matrix is then input into the generator, which adopts three consecutive graph convolution propagation process ("GC1" to "GC3" in Fig. 1, defined by (4)) activated by Exponential Linear Units (ELUs) [44] for feature learning.

In each convolution, the number of graph features are extended from N to 128, 256, and 512, respectively, and node features diffuse to the respective immediate neighborhoods in accordance with A. Using these convolutions, the generator transforms the input partial graph features into an $|\mathcal{R}| \times 512$ intermediate feature matrix. This matrix is later decoded by three other ELU-activated graph convolution layers labeled by "GC4" to "GC6" in Fig. 1, which gradually reduce the number of graph features back to F. The last convolution layer is appended with a sigmoid function $\text{Sigmoid}(x) = (1+e^{-x})^{-1}$ to reconstruct the original graph features with missing values generated during the process. By nature, this feature generator forms an autoencoder, which encodes the input partial data into high-dimensional feature maps and then decodes them to the desired complete graph features.

2) Feature Discrimination: The discriminator aims at distinguishing the generated graph features from the real ones. This network starts with a graph convolution layer, whose output is flattened and fed into two subsequent fully-connected layers. While the graph convolution layer is employed to extract hidden graph features from the input data, the fully-connected neurons in subsequent layers cooperate to classify the extracted features and determine whether the input data is realistic. Finally, a sigmoid function is used at the end to yield a probability between 0 an 1, indicating the discriminator's belief on input authenticity.

C. GCGA Training Method

Before using GCGA for graph feature completion, the network parameter values – $W^{(l)}$ and $b^{(l)}$ in (4), weight and bias matrices in the fully connected layers [23] – need to be properly adjusted. While GCGA extends the adversarial principle of GAN to design the architecture, this model cannot be easily trained with methodologies designed for GAN. The main difference between GCGA and GAN lies in the generator design, which takes partial graph features instead of random noise data as input. Hence, we propose a training method tailored for GCGA to account for the autoencoder nature of the graph feature generator.

Given a collection of C node feature matrices $\{X_{(c)} = \{x_{ij,(c)}\}\}_{c=1}^{C}$, we use sets $\mathcal{R}_{(c)}^{+}$ and $\mathcal{R}_{(c)}^{-}$ to represent the nodes whose features are available and missing in $X_{(c)}$, respectively. We first augment the feature data by artificially removing some of the available features in each $X_{(c)}$ and construct a new data set $\mathcal{X}$. Algorithm 1 presents the pseudo-code for the data augmentation process. Specifically, we first generate M unique $\tilde{\mathcal{R}}_{(c),m}^{-} \subsetneq \mathcal{R}_{(c)}^{+}$ sets for each $X_{(c)}$ (line 4 in Algorithm 1). The available features of all nodes in $\tilde{\mathcal{R}}_{(c),m}^{-}$ are then removed from $X_{(c)}$ to construct a new partial feature matrix $X_{(c),m}^{-} = \{x_{ij,(c),m}^{-}\}$ (lines 3 and 6). As a result, $M \times C$ cases for the feature generator can be constructed, each of which comprises an input matrix $X_{(c),m}^{-}$ and a target output matrix $X_{(c)}$. The objective of the feature generator is to develop an estimate of $X_{(c)}$ using $X_{(c),m}^{-}$ while accurately recover those artificially removed features.

The network parameters in the feature generator and discriminator are adjusted iteratively. In each iteration,

Algorithm 1 Data Augmentation for GCGA Training

Data: $\{X_{(c)}\}_{c=1}^{C}$, M
Result: $\mathcal{X} = \{X_{(c),m}^{-}, X_{(c)}\}_{c=1,m=1}^{C\times M}$
1 initialize an empty augmented feature data set $\mathcal{X}$;
2 **for** $\{c, m\} \in \{1, 2, \cdots, C\} \times \{1, 2, \cdots, M\}$ **do**
3 $X_{(c),m}^{-} \leftarrow X_{(c)}$;
4 generate a random $\tilde{\mathcal{R}}_{(c),m}^{-}$ as a subset of $\mathcal{R}_{(c)}^{+}$;
5 **for** $\{n, r\} \in \tilde{\mathcal{R}}_{(c),m}^{-} \times \{1, 2, \cdots, R\}$ **do**
6 $x_{nr,(c),m}^{-} \leftarrow 0^2$;
7 **end**
8 append $X_{(c),m}^{-}$ to $\mathcal{X}$;
9 **end**

the artificially constructed partial feature matrix $X_{(c),m}^{-}$ in each training case is first autoencoded by the generator, which develops an estimate of $X_{(c)}$ denoted by $\hat{X}_{(c),m}^{-} = \{\hat{x}_{ij,(c),m}^{-}\}$. These estimations are first compared with their corresponding $X_{(c)}$ matrix to calculate the mean square error (MSE) over available features in $X_{(c),m}^{-}$, i.e.,

$$\text{MSE}(\hat{X}_{(c),m}^{-}, X_{(c)}) = \frac{1}{|\tilde{\mathcal{R}}_{(c),m}^{+}|} \sum_{n=1}^{|\tilde{\mathcal{R}}_{(c),m}^{+}|} \sum_{r=1}^{R} |x_{nr,(c)} - \hat{x}_{nr,(c),m}^{-}|^2 \tag{5}$$

where $\tilde{\mathcal{R}}_{(c),m}^{+} = \mathcal{R}_{(c)}^{+} \setminus \tilde{\mathcal{R}}_{(c),m}^{-}$. The MSE loss function for the generator can be subsequently defined as

$$L^{\text{G}} = \frac{1}{CM} \sum_{c=1}^{C} \sum_{m=1}^{M} \text{MSE}(\hat{X}_{(c),m}^{-}, X_{(c)}) \tag{6}$$

After calculating L^{G}, the estimations are subsequently used to adjust θ^{D}. Since the discriminator's role is to distinguish estimations from their target matrix $X_{(c)}$, which has missing features at nodes $\mathcal{R}_{(c)}^{-}$, the features of these nodes are also removed in $\hat{X}_{(c),m}^{-}$ for a fair classification. We assign the objective classification values of $\hat{X}_{(c),m}^{-}$ matrices to zeros, and set those of $X_{(c)}$ to ones. Adopting the binary cross entropy (BCE) loss function:

$$\text{BCE} = -\sum_{n=1}^{N} [y_{(n)} \log \hat{y}_n + (1 - y_{(n)}) \log(1 - \hat{y}_{(n)})] \tag{7}$$

where N is the number of test cases, $y_{(n)}$ is the objective classification value and $\hat{y}_n$ is the actual classification result using the discriminator, the loss values for all $X_{(c),m}^{-}$ and $X_{(c)}$ matrices can be calculated, which are denoted by $L_{\mathbf{Gz}}^{\text{D}}$ and $L_{\mathbf{x}}^{\text{D}}$, respectively. Finally, the aggregated loss value, i.e., $L_{\mathbf{Gz}}^{\text{D}} + L_{\mathbf{x}}^{\text{D}}$ is used as the training objective of the discriminator, whose network parameters are optimized by Adam [45]. Similarly, the generator parameters are adjusted with respect to $1 - L_{\mathbf{Gz}}^{\text{D}} + L^{\text{G}}$ after tuning those of the discriminator. This finishes an iteration of the GCGA training method. The whole training process terminates when the generator cannot further improve the quality of the generated node features.

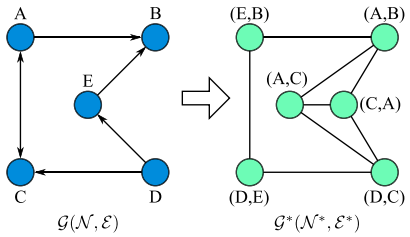


Fig. 2. An example of graph transformation from $\mathcal{G}$ to $\mathcal{G}^*$.

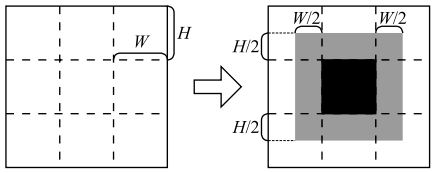


Fig. 3. An example of the sub-region operation in speed estimation. Shadowed area is the padding region of the solid sub-region.

D. GCGA-Based Speed Estimation

In the previous subsections, the model and training method of GCGA are proposed. Nonetheless, GCGA aims at generating realistic node features of a graph, which cannot be directly applied to solve the traffic speed estimation problem presented in Section II. In this subsection, we discuss the implementation of GCGA in estimating traffic speed given incomplete real-time data.

The first and foremost issue is that the problem requires a solution to recover features of graph edges instead of nodes. This issue can be resolved by transforming the original transportation network graph into a new road-connectivity graph. Specifically, we create a new undirected graph $\mathcal{G}^*(\mathcal{N}^*, \mathcal{E}^*)$ based on $\mathcal{G}$, where $\mathcal{N}^* = \{e | e \in \mathcal{E}\}$ is a set of nodes, each of which corresponds to a road in the original network. Obviously, we have $|\mathcal{E}| = |\mathcal{N}^*|$. Furthermore, $\mathcal{E}^*$ denotes the connectivity of roads in $\mathcal{E}$, which is defined as follows:

$$\mathcal{E}^* = \{(e_1, e_2) | \forall e_1, e_2 \in \mathcal{E}, \{\text{fr}(e_1), \text{to}(e_1)\} \cap \{\text{fr}(e_2), \text{to}(e_2)\} \neq \emptyset\}. \quad (8)$$

Fig. 2 gives an example on how the transformation is performed. In the illustration, the labels in the left graph denotes the road intersections in the original transportation network, and those in the right graph are the roads. This graph transformation is incentivized by the idea that the traffic speeds of connected roads are likely correlated but not necessarily similar, which accords with the intuition and analysis in [32].

With the transformed road-based graph, GCGA can be applied to handle the traffic speed estimation problem. The feature generation network, which is a GCGA sub-network, is employed to develop the estimated speed based on sparsely available information with its structure and parameters unchanged. The feature discrimination network, while not directly used in online generating speed maps, helps the feature generation network to develop realistic maps during the training process as introduced in Section III-C. Hence, both sub-networks in GCGA is indispensible in the proposed GCGA-based speed estimation approach.

Given real-time traffic speeds of some roads in the network, one can construct a feature matrix input $X \in \mathbb{R}^{|\mathcal{N}^*| \times 6}$. For each road in the network, the features are defined as 1) its normalized real-time speed measurement (zero if unavailable), 2) maximum allowed traffic speed, 3) length of the road, 4) width of the road, 5) number of lanes, and 6) number of point-of-interests nearby. Utilizing the fine-tuned network parameters, GCGA can generate a complete feature matrix output $Z \in \mathbb{R}^{|\mathcal{N}^*| \times 1}$ with the graph feature generator, in which each value corresponds to the normalized real-time traffic speed between zero and one. One may note that in this work, the features of intersections, i.e., $\mathcal{N}$ in $\mathcal{G}$, is not given like $\mathcal{N}^*$ in $\mathcal{G}^*$ above (which is equivalent to $\mathcal{E}$ in $\mathcal{G}$). Recall the definition of X and (4) in Section III-A2, the computation of GCGA only involves the node features of the input graph, which is $\mathcal{N}^*$ in $\mathcal{G}^*$. None of the intersection features except for the adjacency information (A) is required during the process. Therefore, we do not given a fixed definition on intersection features, and using such features in traffic speed estimation is a potential future research.

The above speed estimation process yields satisfactory accuracy and computation time performance when handling small transportation networks. However, the GCGA training time significantly increases for large urban regions where the road networks are complicated. This is partially contributed to the matrix multiplication calculations in (4), where $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ is a $|\mathcal{N}^*| \times |\mathcal{N}^*|$ matrix. At the same time, the sizes of layer-wise GCN weight parameters in (4), i.e., $W^{(l)}$, are only related to the number of input and output features of the corresponding layer. This implies that the same set of parameters can actually be employed in graphs with different sizes and topologies given that they share similar characteristics. Therefore, in the implementation of GCGA-based speed estimation, the complete investigated area is divided into sub-regions for fast computation.

Fig. 3 gives an illustrative example of this process. In particular, we first divide the whole region into sub-regions of equal size $W \times H$. For each sub-region, a surrounding padding region is formulated as depicted by the shadowed area in Fig. 3, with height $H/2$ and width $W/2$ on each side. We aggregate all roads in the sub-region and the padding region, and construct a sub-graph of the original transportation network. In each time, the features (speed measurements, maximum allowed speeds, etc.) in this sub-graph are input into GCGA for parameter training and speed estimation. While the graph feature generator develops estimations for both regions, only those in the sub-region are considered in the subsequent feature discriminator for training, or the real-time traffic speed in estimation.

IV. Case Studies

In this work, we propose GCGA to address the real-time traffic speed estimation problem. To fully evaluate the performance of the proposed technique, we conducted three comprehensive case studies with a real world dataset. We first investigate the accuracy of estimated speed using the

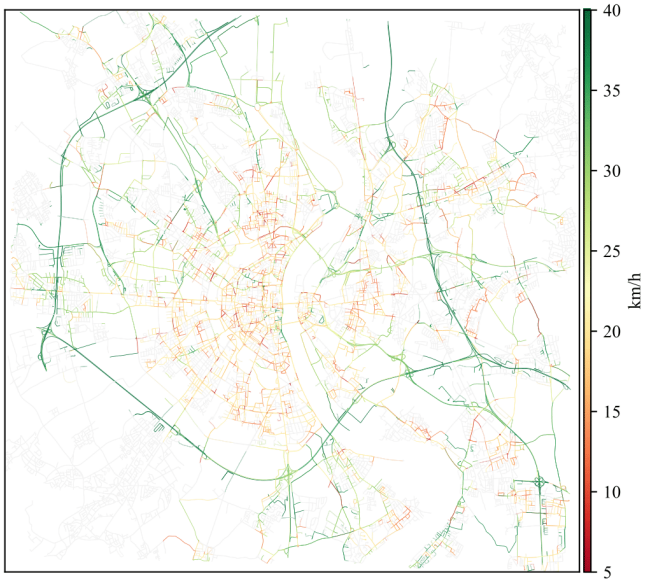


Fig. 4. A 20km-by-20km transportation network of Cologne, Germany obtained from OpenStreetMap [46] colored by the traffic speed from 5:00 to 5:05 in the morning.

technique, and compare the proposed technique with existing solutions. Next, we study how dataset and problem properties influence the estimation accuracy and model training time. Lastly, we assess the impact of control parameters on the system performance.

A. Dataset and Simulation Configurations

In this work, we adopt the real-world traffic data at Cologne, Germany for investigation. Specifically, the transportation network of Cologne is first obtained from OpenStreetMap [46] using OSMnx [47], which is depicted in Fig. 4. In this network, 16658 nodes are sparsely connected and there are 37034 edges. Furthermore, we employ the vehicular mobility trace of Cologne [48], [49] to construct benchmark real-time traffic speed maps of the transportation network. This dataset provides 3.54 billion GPS and speed records of more than 700000 individual vehicle trips for a period of 23 hours (1:00 to 24:00) in a typical working day [48].

To develop the ground truth traffic speed maps, we first split the complete dataset into $23 \times 12 = 276$ chunks, each of which corresponds to the vehicular traces of five minutes in the day. In each timeslot, every record is fitted to its nearest road in the network, and the corresponding driving speed is stored as a speed data point of the road. Specifically, the investigated area is first divided into map grids of 50×50 meters. Each road in the network is further divided into road segments according to OpenStreetMap records, which are straight lines between geographic locations in the map [46]. When fitting a GPS record to the network, we first find out the map grid in which the record resides and its adjacent grid in all directions, resulting in a patch of nine map grids. Then the distances between the record and all road segments in the patch are calculated with trigonometric functions, and we consider the record fits the road which comprises the nearest road segment [50]. After all records in the timeslot are processed, each road calculates the average value of all stored speed data points, which is considered as the traffic speed of the road at the specific five-minute interval. If there is no speed data points for an arbitrary road, the traffic speed is considered unavailable in the ground truth data. Fig. 4 presents an example of the traffic speed map from 5:00 to 5:05 in the morning, which is developed using the method above. Note that roads without vehicle traces in the specified time is plotted with gray color.

To emulate real-world cases in which the stationary speed sensors and crowd-sourced vehicular GPS records can only provide the traffic speed of a small portion of all roads, we randomly[3] remove speed values from speed maps to construct GCGA training and estimation cases. In particular, we randomly select 37034α roads whose speeds are retained in the new case, where α defines the data retention rate. All other available speed data are removed. For each of the 276 speed maps, this process is repeated 100 times, resulting in 27600 random cases for GCGA. For cross-validation, these cases are then grouped into three non-overlapping categories, i.e., a training dataset with 13800 cases for adjusting the network parameters, a validation dataset with 6900 cases for stopping the training process, and a testing dataset with 6900 cases for performance assessment. The validation dataset is evaluated after each training epoch, and the training process is terminated if MAPE of the validation cases does not decrease for a consecutive three epoch. When testing the system performance, the testing cases are input into the graph feature generator of GCGA, and the resulting traffic speed estimations are compared with the corresponding available ground truth speed map in the original 276 timeslots using MAPE.

In the following case studies, the whole Cologne transportation network depicted in Fig. 4 is divided into $4 \times 4 = 16$ equal size grids as elaborated in Section III-D. When training GCGA, the control parameter M is set to 24. The sensitivity of these parameters will be investigated in Section IV-D. Furthermore, α is set to 15% unless otherwise stated. The proposed GCGA is modeled with PyTorch [51], and all simulations are conducted on a computing server with two Intel Xeon E5 CPUs, and nVidia GTX 1080 Ti GPUs are employed for neural network computing acceleration.

B. Accuracy of Estimated Traffic Speed

The accuracy of the estimated speed is among the most important metric in evaluating the efficacy of traffic estimation methods. We first investigate the accuracy of estimated real-time traffic speed with the Cologne dataset. In this test, the MAPE performance metric defined in (2) is adopted to evaluate the accuracy.

1) Comparison With Existing Traffic Estimation Approaches: We first compare the performance of the proposed GCGA-based speed estimation method with baseline approaches as follows:

[3]Unless otherwise stated, all random number generators in this work follow the uniform distribution.

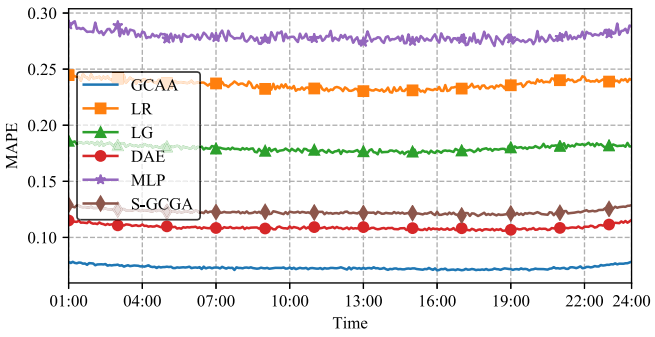


Fig. 5. MAPE of traffic estimation methods on the traffic speed values in Cologne.

- *Linear Regression (LR)* utilizes the road properties and available historical speed data to learn a linear model and estimate traffic speed [32].
- *Linear Regression with Graph Regularization (LG)* [52] models the speed estimation problem as a graph learning problem. The model makes an implicit assumption that connecting roads share similar traffic speeds [32], [52].

We implement LR and LG and use the same simulation configuration as the proposed GCGA-based method to assess their performance. In addition, we adopt the simulation results of the following recent traffic speed estimation approaches from [32] for reference, which employs different datasets from the Cologne one:

- *Matrix Factorization-based Traffic Estimation (MFTE)* [30] employs partial vehicle driving trajectories and meterology context as input data and performs matrix factorization to provide estimates on missing values in a feature matrix.
- *Crowdsourcing-based Traffic Estimation (CTE)* [32] utilizes a graph model to infer the traffic trend in the transportation network and adopts a probabilistic model to learn the traffic speed.

In the comparison, the testing dataset with 6900 cases is employed to evaluate the performance of the compared algorithms, which is identical to GCGA. All other simulation configurations are identical to those stated in Section IV-A for a fair comparison.

The MAPE of all implemented traffic estimation methods over the tested 23 hours are presented in Fig. 5. In this figure, the MAPE of estimated speed values developed by GCGA, LR, and LG on all 276 speed maps are presented. From the results, it is clear that the proposed GCGA-based traffic speed estimation method significantly outperform other compared approaches. On average, GCGA achieves a satisfactory MAPE at 7.3%, while the baseline approaches score 23.6% and 17.9% by LR and LG, MFTE, respectively. Furthermore, reference [32, Sec. VI-C] reports that MFTE and CTE achieve 20%–22% and 10%–15% MAPE, respectively. The outstanding performance of GCGA is contributed by its unique generative model design. In addition, the deep GCN architecture in the graph feature generator can better extract spatial characteristics of the partial information provided as input.

For a more direct view of how well GCGA can estimate the traffic speed, Fig. 6 presents an example of the input traffic speed map and the generated data by the proposed GCGA-based method at 5:00–5:05 and 7:30–7:35 in the morning. While the former case represents a typical transportation condition of the network, the latter is among the most congested time in the working day. From the results we can observe that the central downtown of Cologne is the most congested region in the city. The congested area significantly grows from 5:00 to 7:30 in the morning, which corresponds to the massive volume of daily commutes. Nonetheless, the MAPE does not demonstrate obvious spatial or temporal correlation with respect to the central congestion condition. In addition, we summarize the MAPE histogram of traffic speeds in Fig. 7. From the figure, no clear relation between MAPE and averaged traffic speed can be established. This indicates that GCGA can develop robust traffic speed estimations on different traffic conditions.

2) Comparison With Deep Learning Approaches: Besides the previously compared traffic estimation approaches in the literature, there are also some other deep learning approaches that can be employ to address data imputation tasks. We compare the proposed GCGA with the following deep learning methods:

- *Deep Autoencoder (DAE)* [53] learns an encode of a set of input data with deep neural networks. In this work, we use the graph feature generater part of GCGA as the compared graph-generating DAE, which is trained with respect to L^{G}.
- *Multi-hidden-layer Multilayer Perceptron Network (MLP)* [23] is a class of neural networks with multiple hidden layers. In this work, the tested MLP is designed with the same architecture as illustrated in Fig. 1, but with each of the graph convolution computation replaced by fully connected links.
- *Shallow GCGA (S-GCGA)* is a variant of the proposed GCGA but with a "shallower" architecture. In particular, two graph convolution layers, namely "GC3" and "GC4", are removed from GCGA to formulate S-GCGA.

All compared deep learning techniques are trained using Adam optimizer with the same config as GCGA. All methods share the same training, validation, and testing dataset, and all other simulation configurations are identical to previous tests. The MAPE of all compared techniques are also depicted in Fig. 5. Compared with the other deep learning approaches, the proposed GCGA can still maintain its leading position. DAE, MLP, and S-GCGA score 10.9%, 27.9%, and 12.2%, respectively. This result indicates that the basic constituting components of GCGA, i.e., graph convolution, generative adversarial model, and deep neural network architecture, are all critical to the satisfactory accuracy. Since DAE lacks the discriminator and S-GCGA has a shallower architecture, their capability of generating realistic data and feature extraction is undermined. This demonstrates the efficacy of the GAN-like generator-discriminator design in GCGA. Meanwhile, MLP performs badly in the test. This is because the adjacency information of a graph is critical to problems such as traffic speed estimation. However, the information is not included in the input data of MLP, which has to learn it during the training process.

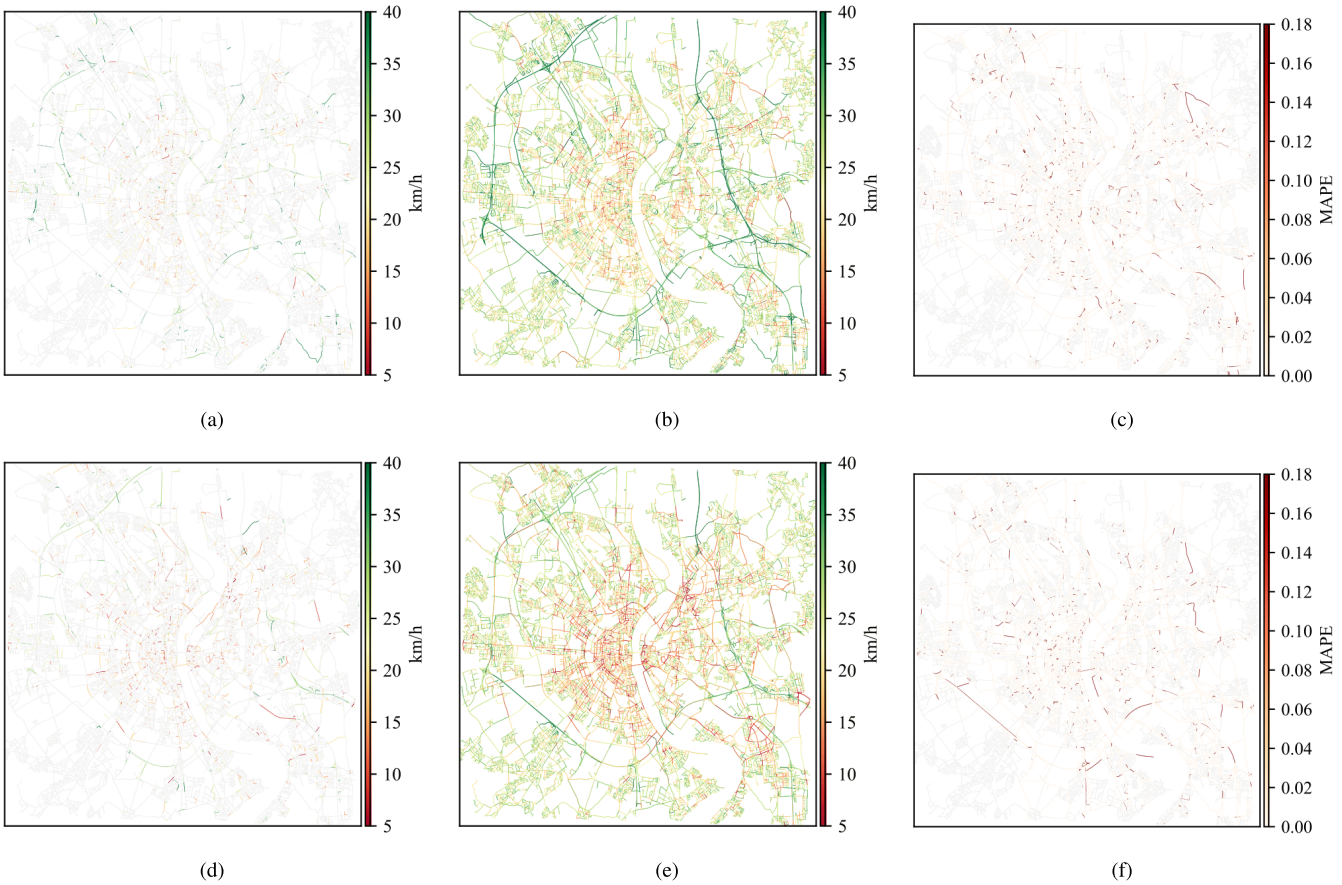


Fig. 6. An example input speed map and the generated traffic speed map from 5:00 to 5:05 and from 7:30 to 7:35 by GCGA. (a) Input traffic speed at 5:00 to 5:05. (b) Generated traffic speed at 5:00 to 5:05. (c) MAPE from ground truth in Fig. 4 at 5:00 to 5:05. (d) Input traffic speed at 7:30 to 7:35. (e) Generated traffic speed at 7:30 to 7:35. (f) MAPE from ground truth at 7:30 to 7:35.

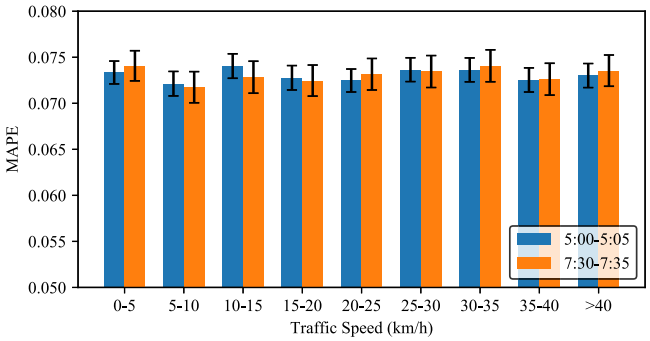


Fig. 7. MAPE histogram of traffic speeds.

Last but not least, the proposed GCGA-based traffic speed estimation method requires less than one second to produce the complete traffic speed map given fine-tuned GCGA network parameters. Considering that the speed map is updated every five minutes in the testing dataset, this suggest that the proposed method can develop accurate speed estimations in real-time.

C. Impact of Dataset Properties

In the adopted Cologne dataset, we aggregate the vehicular trace records within five minutes to develop each of the traffic speed maps. Furthermore, the data retention rate is artificially set to 15%. In practice, different transportation networks may require real-time traffic speed estimations with various temporal resolutions, and the real retention rate is closely related to the number of stationary sensors and crowdsourcing vehicles. Therefore, it can help us understand the robustness of the proposed GCGA-based speed estimation method by assessing the impact of these properties.

In this subsection we first perform a parameter sweep test on the dataset temporal resolution. In particular, the adopted Cologne dataset is re-processed using the same method elaborated in Section IV-A, and each traffic speed map comprises the vehicular trace records in 30 seconds, one minute, ten minutes, and 30 minutes for four new test scenarios, respectively. We are interested in its impact on the estimation accuracy.

The MAPE of GCGA with different temporal resolutions are depicted in Fig. 8. From this plot, it is clear that the estimation accuracy gradually increases with the decrease of temporal resolution at any time of a day. Furthermore, when the vehicular traces with more than five minutes are aggregated, the performance of GCGA is quite similar from 7:00 to 20:00. This observation accords with the intuition that a higher resolution can lead to a more fluctuating traffic speed for a same road, since the influence of each individual

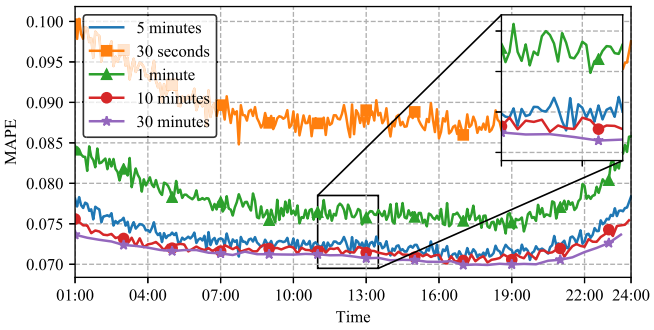


Fig. 8. MAPE of GCGA with different temporal resolutions.

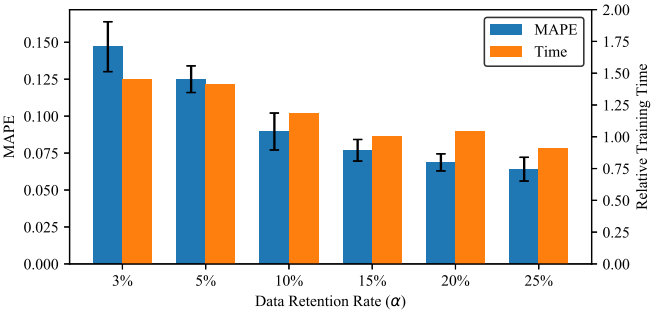


Fig. 9. MAPE and training time of GCGA with different data retention rate. The relative training time of $\alpha = 15\%$ scenario is one, which corresponds to approximately 17 hours on one testing GPU.

vehicle is emphasized with the reduction of total number of traces. In daytime and peak hours, the large amount of traces make the sample mean of speed values from traces better resemble the population mean, resulting in a more stable performance. Both sample size and traffic (flow) dynamics can greatly impact the sample variances of trace speeds in the dataset. Nonetheless, as GCGA receives sample means as inputs instead of individual observations, sample variances do not play a significant role in generating accurate speed estimations. On the other hand, the vehicle trace data during night-time and off-peak hours, e.g., 22:00 to 3:00, is sparse with high temporal resolutions ($\leq$5 minutes). This makes it difficult for the system to develop highly accurate estimations.

Besides the temporal resolution, another critical dataset property that may influence the system performance is the data retention rate α. In the previous test, we assume that $\alpha = 15\%$, which accords with most of the case studies in [32]. Nonetheless, it is possible that real-world datasets may have more or less roads whose traffic speed is known. In this test, we assess the performance of GCGA with $\alpha \in \{3\%, 5\%, 10\%, 20\%, 25\%\}$. All other simulation configurations are identical to the settings in Section IV-B. The results are summarized in Fig. 9. From the figure we can conclude that GCGA can better address datasets with more available traffic speed values, i.e., larger α. Such datasets can provide more inter-correlated traffic information in the input training cases, which can facilitate GCGA to extract the spatial features in the speed map better and faster. In the meantime, even with only 3% of the speed data available, the proposed system can still generate complete speed maps with less than 15% MAPE

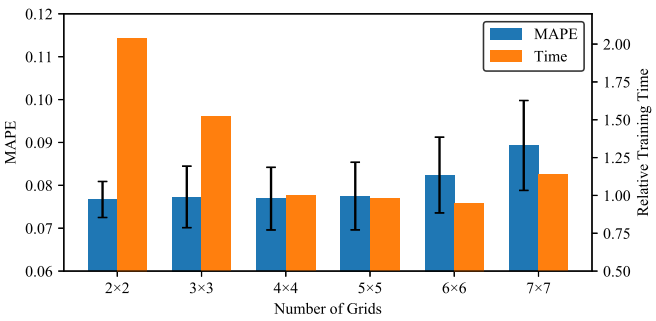


Fig. 10. MAPE and training time of GCGA with different grid size. The relative training time of 16-grid scenario is one.

on average. While such speed values are not as accurate, they can still demonstrate a rough picture on the status for the investigated transportation system.

D. Parameter Sensitivity Analysis

In previous simulations, we divide the whole Cologne transportation network shown in Fig. 4 into 16 grids to reduce GCGA training time, and $M = 24$ is employed in the data augmentation process of the training. In this subsection, we investigate the system sensitivity on these two control parameters. Specifically, we first divide the network into $2 \times 2 = 4$, $3 \times 3 = 9$, $5 \times 5 = 25$, $6 \times 6 = 36$, and $7 \times 7 = 49$ grids to re-train GCGA and assess the performance deviation from the baseline 16-grid scenario. The results are summarized in Fig. 10. From the figure it is clear that the grid size does not significantly influence the accuracy of estimated traffic speed values as long as the grid size is sufficiently large with respect to the investigated area. Nonetheless, too small grids, e.g., 7×7 case, can only provide partial spatial relationship of the network, rendering worse MAPE. In the meantime, larger grid generally yields longer training time, which accords with our previous analysis in Section III-D. The training time for 16, 25, and 36 grids is almost the same. This is because while the matrix multiplication in (4) is accelerated with a smaller matrix size, the total number of test cases also increases with the grids. When the number of grids increases to 49, the increase in total number of testing cases and training difficulty overwhelm the decrease in matrix computation complexity. Hence the training time increases. To conclude, dividing the Cologne dataset into 16 grids, each of which on average contains approx. 2000 roads, for GCGA is preferred in terms of model training time.

Finally, we test the sensitivity of parameter M with values in $\{4, 9, 49\}$. Combining with the baseline test in which $M = 24$, the results are summarized in Fig. 11. From the figure it can be observed that increasing M can lead to better estimation accuracy, but the training process is also lengthened. In addition, when M is greater than 24, the accuracy improvement is not notable enough to compensate the significantly increased training time. In GCGA, the more adversarial traffic maps generated by the graph feature generator, the better it can be trained to circumvent the detection of the discriminator. On the other hand, the traffic map classification capability of the discriminator is limited by its neural

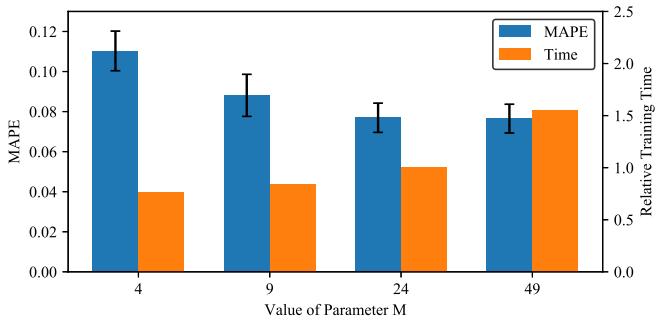


Fig. 11. MAPE and training time of GCGA with different M. The relative training time of $M = 24$ scenario is one.

network architecture. This explains why the estimation accuracy cannot be further improved when more than sufficient adversarial data are developed.

V. Conclusions

In this work, we propose a new deep learning model called GCGA to address the real-time traffic speed estimation problem. Different from existing approaches, the proposed technique can extract the spatial characteristics from the transportation network and construct the complete traffic speed map with partial measurements using a generative learning model. The proposed GCGA incorporates the concept and design principle from GCN and GAN to provide an outstanding graph feature completion capability. In addition, due to the incomplete graph input data, we propose a practical GCGA training method to fine-tune the network parameters. Furthermore, the proposed GCGA is adopted to address the traffic speed estimation problem, and the issues in its implementation are addressed.

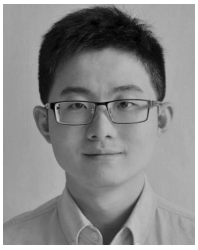
We conduct comprehensive case studies to assess the performance of the proposed approach utilizing the transportation network and vehicular trace records in Cologne, Germany. The simulation results demonstrate a satisfactory estimation accuracy for GCGA. Additionally, GCGA can outperform both existing traffic speed estimation solutions and other representative deep learning approaches in addressing this problem. We also analyze the impact of the dataset properties with simulations, and illustrate the sensitivity of GCGA with respect to various control parameter configurations.

The future work can go in three directions. First, it is possible to extend the existing architecture to address traffic flow estimation/prediction and speed prediction problems by incorporating deep learning prediction techniques. Existing research [19] has demonstrated that deep learning approaches can handle the problems with efficacy, and the proposed GCGA can be further integrated with recurrent neural networks to provide temporal inference capability for predictions. Second, the plethora of traffic-model-based methods for traffic speed estimation as introduced in Section I may help design a hybridized speed estimator based on GCGA, which can be expected to inherit merits from both parents. Finally, GCGA is proposed as a general-purpose graph feature completion approach. It is promising to apply the methodology in solving other research and industrial problems, e.g., [54].

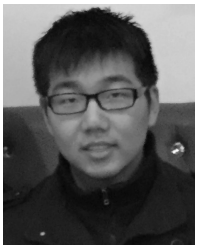
References

[1] F.-Y. Wang, "Parallel control and management for intelligent transportation systems: Concepts, architectures, and applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 630–638, Sep. 2010.
[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.
[3] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1624–1639, Dec. 2011.
[4] A. Y. Saber and G. K. Venayagamoorthy, "Plug-in vehicles and renewable energy sources for cost and emission reductions," *IEEE Trans. Ind. Electron.*, vol. 58, no. 4, pp. 1229–1238, Apr. 2011.
[5] J. Q. James and A. Y. S. Lam, "Autonomous vehicle logistic system: Joint routing and charging strategy," *IEEE Trans. Intell. Transp.*, vol. 19, no. 7, pp. 2175–2187, Jul. 2018. doi: 10.1109/TITS.2017.2766682.
[6] J. J. Q. Yu, "Two-stage request scheduling for autonomous vehicle logistic system," *IEEE Trans. Intell. Transp. Syst.*, to be published.
[7] M. Keyvan-Ekbatani, M. Yildirimoglu, N. Geroliminis, and M. Papageorgiou, "Multiple concentric gating traffic control in large-scale urban networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 2141–2154, Aug. 2015.
[8] R. Sundar, S. Hebbar, and V. Golla, "Implementing intelligent traffic control system for congestion control, ambulance clearance, and stolen vehicle detection," *IEEE Sensors J.*, vol. 15, no. 2, pp. 1109–1113, Feb. 2015.
[9] Y. Duan, Y. Lv, Y.-L. Liu, and F. Wang, "An efficient realization of deep learning for traffic data imputation," *Transp. Res. C, Emerg. Technol.*, vol. 72, pp. 168–181, Nov. 2016.
[10] S. Taghvaeeyan and R. Rajamani, "Portable roadside sensors for vehicle counting, classification, and speed measurement," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 1, pp. 73–83, Feb. 2014.
[11] J. J. Q. Yu, A. Y. S. Lam, and Z. Lu, "Double auction-based pricing mechanism for autonomous vehicle public transportation system," *IEEE Trans. Intell. Veh.*, vol. 3, no. 2, pp. 151–162, Jun. 2018.
[12] J. J. Q. Yu and A. Y. S. Lam, "Core-selecting auctions for autonomous vehicle public transportation system," *IEEE Syst. J.*, to be published.
[13] (Feb. 2018). *Official Google Blog: The Bright Side of Sitting in Traffic: Crowdsourcing Road Congestion Data.* [Online]. Available: https://googleblog.blogspot.hk/2009/08/bright-side-of-sitting-in-traffic.html
[14] L. Li, X. Su, Y. Zhang, Y. Lin, and Z. Li, "Trend modeling for traffic time series analysis: An integrated study," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3430–3439, Dec. 2015.
[15] L. Li, Y. Li, and Z. Li, "Efficient missing data imputing for traffic flow by considering temporal and spatial dependence," *Transp. Res. C, Emerg. Technol.*, vol. 34, pp. 108–120, Sep. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X13001095
[16] R. Wang, Y. Li, and D. B. Work, "Comparing traffic state estimators for mixed human and automated traffic flows," *Transp. Res. C, Emerg. Technol.*, vol. 78, pp. 95–110, May 2017.
[17] R. Wang, D. B. Work, and R. Sowers, "Multiple model particle filter for traffic estimation and incident detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 12, pp. 3461–3470, Dec. 2016.
[18] C. P. I. J. van Hinsbergen, T. Schreiter, F. S. Zuurbier, J. W. C. van Lint, and H. J. van Zuylen, "Localized extended Kalman filter for scalable real-time traffic state estimation," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 385–394, Mar. 2012.
[19] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.
[20] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, May 2018, pp. 1–33.
[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, Toulon, France, Apr. 2017, pp. 1–14.
[22] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2014, pp. 2672–2680.
[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, F. Bach, Ed. Cambridge, MA, USA: MIT Press, Nov. 2016.
[24] M. Levin and Y.-D. Tsao, "On forecasting freeway occupancies and volumes (abridgment)," *Transp. Res. Rec.*, no. 773, pp. 47–49, Oct. 1980.
[25] S. Lee and D. B. Fambro, "Application of subset autoregressive integrated moving average model for short-term freeway traffic volumes forecasting," *Transp. Res. Rec.*, nos. 16–78, pp. 179–188, Jan. 1999.

[26] B. Ghosh, B. Basu, and M. O'Mahony, "Multivariate short-term traffic flow forecasting using time-series analysis," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 2, pp. 246–254, Jun. 2009.
[27] M. Lippi, M. Bertini, and P. Frasconi, "Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 2, pp. 871–882, Jun. 2013.
[28] Y. Zhu, Z. Li, H. Zhu, M. Li, and Q. Zhang, "A compressive sensing approach to urban traffic estimation with probe vehicles," *IEEE Trans. Mobile Comput.*, vol. 12, no. 11, pp. 2289–2302, Nov. 2013.
[29] H. Chang, Y. Lee, B. Yoon, and S. Baek, "Dynamic near-term traffic flow prediction: System-oriented approach based on past experiences," *IET Intell. Transp. Syst.*, vol. 6, no. 3, pp. 292–305, Sep. 2012.
[30] J. Shang, Y. Zheng, W. Tong, E. Chang, and Y. Yu, "Inferring gas consumption and pollution emission of vehicles throughout a city," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, Aug. 2014, pp. 1027–1036.
[31] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 25–34.
[32] H. Hu, G. Li, Z. Bao, Y. Cui, and J. Feng, "Crowdsourcing-based real-time urban traffic speed estimation: From trends to speeds," in *Proc. 32nd IEEE Int. Conf. Data Eng.*, Helsinki, Finland, May 2016, pp. 883–894.
[33] Y. Lv, Y. Chen, X. Zhang, Y. Duan, and N. Li, "Social media based transportation research: The state of the work and the networking," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 1, pp. 19–26, Jan. 2017.
[34] Y.-S. Jeong, Y.-J. Byon, M. Mendonca Castro-Neto, and S. M. Easa, "Supervised weighting-online learning algorithm for short-term traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 4, pp. 1700–1707, Dec. 2013.
[35] K. Y. Chan, T. S. Dillon, J. Singh, and E. Chang, "Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and Levenberg–Marquardt algorithm," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 644–654, Jun. 2012.
[36] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma. (2017). "Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks." [Online]. Available: https://arxiv.org/abs/1705.02699
[37] X. Cheng, R. Zhang, J. Zhou, and W. Xu. (2017). "DeepTransport: learning spatial-temporal dependency for traffic condition forecasting." [Online]. Available: https://arxiv.org/abs/1709.09585
[38] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, Jan. 2018.
[39] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, "Generative adversarial networks: Introduction and outlook," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 4, pp. 588–598, Sep. 2017.
[40] I. Goodfellow. (2016). "NIPS 2016 tutorial: Generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1701.00160
[41] D. Duvenaud *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, Dec. 2015, pp. 2224–2232.
[42] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," in *Proc. Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, May 2018, pp. 1–15.
[43] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 3844–3852.
[44] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *Proc. Int. Conf. Learn. Represent.*, San Juan, PR, USA, May 2016, pp. 1–14.
[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, San Diego, CA, USA, Dec. 2015, pp. 1–41.
[46] (Dec. 2017). *OpenStreetMap*. [Online]. Available: http://www.openstreetmap.org/
[47] G. Boeing, "OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Comput. Environ. Urban Syst.*, vol. 65, pp. 126–139, Sep. 2016.
[48] (Dec. 2017). *Vehicular Mobility Trace of the City of Cologne Germany*. [Online]. Available: http://kolntrace.project.citi-lab.fr/#trace
[49] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and analysis of a large-scale urban vehicular mobility dataset," *IEEE Trans. Mobile Comput.*, vol. 13, no. 5, pp. 1061–1075, May 2014.
[50] J. J. Q. Yu, W. Yu, and J. Gu, "Online vehicle routing with neural combinatorial optimization and deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, to be published.
[51] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 1–4.
[52] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semi-supervised learning on large graphs," in *Proc. Int. Conf. Comput. Learn. Theory*. Banff, AB, Canada, Jul. 2004, pp. 624–638.
[53] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[54] J. J. Q. Yu, A. Y. S. Lam, D. J. Hill, Y. Hou, and V. O. K. Li, "Delay aware power system synchrophasor recovery and prediction framework," *IEEE Trans. Smart Grid*, to be published.



James Jian Qiao Yu (S'11–M'15) received the B.Eng. and Ph.D. degrees in electrical and electronic engineering from The University of Hong Kong, Hong Kong, in 2011 and 2015, respectively, where he was a Post-Doctoral Fellow from 2015 to 2018. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China, and an Honorary Assistant Professor with the Department of Electrical and Electronic Engineering, The University of Hong Kong. He is also the Chief Research Consultant of GWGrid Inc., Zhuhai, and Fano Labs, Hong Kong. His research interests include smart city technologies, deep learning and big data, intelligent transportation systems, and energy systems.



Jiatao Gu received the B.Eng. degree from Tsinghua University in 2014. He is currently pursuing the Ph.D. degree in electrical and electronic engineering with The Hong Kong University. In 2018, he joined Facebook Artificial Intelligence Research. He was a Research Intern with Salesforce Research and Microsoft Research, and a Visiting Student with New York University. He has published papers in top computer science conferences, e.g., ACL, EMNLP, NAACL, EACL, AAAI, and ICLR. His research interests include natural language processing and deep learning, especially on neural machine translation.