How to Build a Graph-Based Deep Learning Architecture in Traffic Domain: A Survey

Jiexia Ye, Juanjuan Zhao, *Member, IEEE*, Kejiang Ye, *Member, IEEE*, and Chengzhong Xu, *Fellow, IEEE*

Abstract— **In recent years, various deep learning architectures have been proposed to solve complex challenges (e.g. spatial dependency, temporal dependency) in traffic domain, which have achieved satisfactory performance. These architectures are composed of multiple deep learning techniques in order to tackle various challenges in traffic tasks. Traditionally, convolution neural networks (CNNs) are utilized to model spatial dependency by decomposing the traffic network as grids. However, many traffic networks are graph-structured in nature. In order to utilize such spatial information fully, it's more appropriate to formulate traffic networks as graphs mathematically. Recently, various novel deep learning techniques have been developed to process graph data, called graph neural networks (GNNs). More and more works combine GNNs with other deep learning techniques to construct an architecture dealing with various challenges in a complex traffic task, where GNNs are responsible for extracting spatial correlations in traffic network. These graph-based architectures have achieved state-of-the-art performance. To provide a comprehensive and clear picture of such emerging trend, this survey carefully examines various graph-based deep learning architectures in many traffic applications. We first give guidelines to formulate a traffic problem based on graph and construct graphs from various kinds of traffic datasets. Then we decompose these graph-based architectures to discuss their shared deep learning techniques, clarifying the utilization of each technique in traffic tasks. What's more, we summarize some common traffic challenges and the corresponding graph-based deep learning solutions to each challenge. Finally, we provide benchmark datasets, open source codes and future research directions in this rapidly growing field.**

Index Terms— **Graph neural networks, GNNs, graph convolution network, GCN, graph, deep learning, traffic forecasting, traffic domain, ITS.**

I. Introduction

ALONG with the acceleration of urbanization process, mass population is quickly gathering together towards cities. In many cities, especially cities in developing countries, the rapidly increasing number of private vehicles and growing demand of public transport services are putting great pressure on their current transportation systems. The traffic problems such as frequent traffic jams, serious traffic accidents and long commute have seriously decreased the operation efficiency of cities and degraded the travel experience of passengers. To address these challenges, many cities are committed to develop an Intelligent Transportation System (ITS) which can provide efficient traffic management, accurate traffic resources allocation and high-quality transportation service. Such a system can reduce traffic accidents, relieve traffic congestion and ensure public traffic safety.

To construct an Intelligent Transportation System which makes cities smart, there are mainly two indispensable components, i.e. intelligent infrastructures and advanced algorithms.

On one hand, with the increasing investment in transportation infrastructures, there are more and more traffic equipments and systems, including loop detectors, probes, cameras on road networks, GPS in taxis or buses, smart cards on subways and buses, automatic fare collection system and online ride-hailing system. These infrastructures produce traffic data around-the-clock, which are heterogeneous data, including numeric data (e.g. GPS trajectories, traffic measurements), image/video data (e.g. vehicle images) and textual data (e.g. incident reports). These transportation data are enormous in volume and complicated in structure, containing complex traffic patterns (e.g. spatiotemporal dependency, highly nonlinearity, complex dynamics). There is an urgent need to utilize more intelligent and powerful approaches to process such traffic data.

On the other hand, in transportation domain, researchers have witnessed the algorithms evolving from statistical methods, to machine learning models and recently to deep learning approaches. In the early stage, statistic methods including ARIMA and its variants [1], [2], VAR [3], Kalman filtering [4] were prevalent, as they have solid and widely accepted mathematical foundations. However, the linear and stationarity assumptions of these methods are violated by the highly non-linearity and dynamics in traffic data, resulting in poor

Manuscript received June 12, 2020; revised October 11, 2020 and November 24, 2020; accepted December 3, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102100, in part by the National Natural Science Foundation of China under Grant 61802387, in part by the China's Post-Doctoral Science Fund under Grant 2019M663183, in part by the Shenzhen Basic Research Program under Grant JCYJ20190812153212464 and Grant JCYJ20170818153016513, in part by the Shenzhen Engineering Research Center for Beidou Positioning Service Improvement Technology under Grant XMHT20190101035, in part by the Science and Technology Development Fund of Macao S.A.R. (FDCT) under Grant 0015/2019/AKP, in part by the Shenzhen Discipline Construction Project for Urban Computing and Data Intelligence, and in part by the Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant 2019349. The Associate Editor for this article was S. Kong. *(Corresponding author: Juanjuan Zhao.)*

Jiexia Ye is with the Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences, Beijing 100049, China, and also with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: jx.ye@siat.ac.cn).

Juanjuan Zhao and Kejiang Ye are with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China (e-mail: jj.zhao@siat.ac.cn; kj.ye@siat.ac.cn).

Chengzhong Xu is with the State Key Laboratory of IOTSC, Department of Computer Science, University of Macau, Taipa, Macau (e-mail: czxu@um.edu.mo).

Digital Object Identifier 10.1109/TITS.2020.3043250

performance in practice. Traditional machine learning approaches such as Support Vector Machine [5], K-Nearest Neighbors [6] can model non-linearity and extract more complex correlations in traffic data. However, the shallow architecture, manual feature selection and separated learning in these models are considered to be unsatisfactory in big data scenarios [7].

The breakthrough of deep learning in many domains, including computer vision, natural language processing has attracted attention from transportation industry and research community. Deep learning techniques overcome the handcrafted feature engineering by providing an end-to-end learning from raw traffic data. The powerful capacities of deep learning techniques to approximate any complex functions in theory can model more complicated patterns in various traffic tasks. In recent years, due to the increasing computing power (e.g. GPU) and sufficient traffic data [7], deep learning based techniques have been widely employed and achieved state-of-the-art performance in various traffic applications. The Recurrent neural networks (RNNs) and Convolutional neural networks (CNNs) based architectures used to be popular in extracting spatiotemporal dependencies. In these architectures, RNN or its variants are employed to extract the temporal correlations in traffic data [8]. CNNs are used to capture the spatial correlations in grid-based traffic network [9]. However, many traffic networks are graph-structured in nature, e.g. road network [10] and subway network. The spatial features learned in CNN are not optimal for representing the graph-based traffic network. Although some previous works have analyzed traffic problems in a graph view [11], [12], these traditional approaches are not powerful enough to process big data and tackle complicated correlations in traffic network.

Recently, many researchers have extended deep learning approaches on graph data to exploit graph structure information [13] and proposed a new group of neural networks called graph neural networks (GNNs) [14], [15], [16], which aims to address graph-related applications. GNNs have become the state-of-the-art approaches in many domains, including computer vision [17], natural language processing [18], biology [19], finance [20], recommendation system [21]. Since many traffic data are graph-structured, many existing works incorporate GNNs into a deep learning architecture to capture the spatial dependency. Recent works have shown that such GNNs-based architectures can achieve better performance than CNNs-based architectures, for that most traffic networks are graph-structured naturally and GNNs can extract the spatial dependency more accurately. In addition, some tasks inherently require researchers to conduct prediction based on a graph, e.g. prediction in traffic network with irregular shapes. Many related works have been produced during the last couple of years and more are on the road. Under this circumstance, a comprehensive literature review on these graph-based deep learning architectures in transportation domain would be very timely, which is exactly our work.

To our best knowledge, we are the first to provide a comprehensive survey on graph-based deep learning works in traffic domain. Note that some works we review actually work on similar traffic problems with similar techniques. Our work can help the upcoming researchers avoid repetitive works and focus on new solutions. What's more, the practical and clear guidance in this survey enables participators to apply these new emerging approaches in real-world traffic tasks quickly.

To sum up, the main contributions of this paper are as follows:

- We systematically outline traffic problems, related research directions, challenges and techniques in traffic domain, which can help related researchers to locate or expand their researches.
- We summarize a general formulation about various traffic problems and provide a specific guidance to construct graphs from several typical kinds of raw traffic datasets. Such thorough summarization is quite practical and can accelerate the applications of graph-based approaches in traffic domain.
- We provide a comprehensive review over typical deep learning techniques widely used in graph-based traffic works. We elaborate their theoretical aspects, advantages, limitations and variants in specific traffic tasks, hoping to inspire the followers to develop more novel models.
- We discuss some challenges shared by most graph-based traffic tasks. For each challenge, we conclude multiple deep learning-based solutions and make necessary comparison, providing useful suggestions for model selection in traffic tasks.
- We collect benchmark datasets, open-source codes in related papers to facilitate baseline experiments in traffic domain. Finally, we propose some future research directions.

The rest of the paper is organized as follows. Section II presents some surveys in traffic domain and some reviews about graph neural networks. Section III briefly outlines several traffic problems and the corresponding research directions, challenges and solutions. Section IV summarizes a general formulation about traffic problems and the graph construction from traffic datasets. Section V analyzes the functionality, advantages and defects of GNNs and other deep learning techniques, as well as examining the tricks to create novel variants of these techniques in specific traffic tasks. Section VI discusses common challenges in traffic domain and the corresponding multiple solutions. Section VII provides hyperlinks of datasets and open codes in papers we investigate. Section VIII presents future directions. Section IX concludes the paper.

II. Related Surveys

There have been some surveys summarizing the development process of algorithms in traffic tasks from different perspectives. Karlaftis *et al.* [22] discussed differences and similarities between statistical methods and neural networks to promote the comprehension between these two communities. Vlahogianni *et al.* [23] reviewed ten challenges on short-term traffic forecasting, which stemmed from the changing needs of ITS applications. Xie *et al.* [24] conducted a comprehensive overview of approaches in urban flow forecasting. Liu *et al.* [7] classified deep learning based urban big data fusion methods into three categories, i.e. DL-output-based fusion,

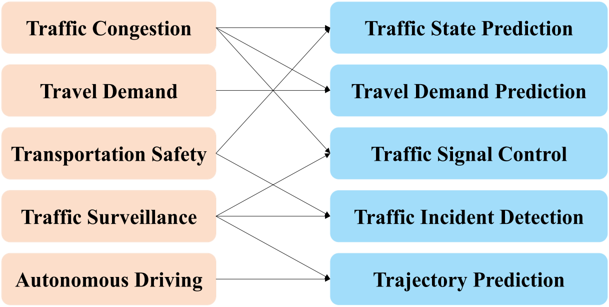


Fig. 1. Typical traffic problems and the corresponding research directions.

DL-input-based fusion and DL-double-stage-based fusion. Deep learning approaches for popular topics including traffic network representation, traffic flow forecasting, traffic signal control, automatic vehicle detection are discussed in [25], [26]. Veres *et al.* [27] and Chen *et al.* [28] gave a similar but more elaborate analysis on new emerging deep learning models in various transportation topics. Wang *et al.* [29] provided a spatial-temporal perspective to summarize deep learning techniques in traffic domain and other domains. However, all these surveys do not take graph neural networks (GNNs) related literatures into consideration, except that Wang *et al.* [29] mentioned GNNs but in a very short subsection.

On the other hand, in recent years, there are several reviews summarizing literatures about GNNs in different aspects. Bronstein *et al.* [30] is the first to overview deep learning techniques on processing data in non-Euclidean space (e.g. graph data). Zhou *et al.* [31] categorized GNNs into graph types, propagation types and training types. In addition, they divided related applications into structural scenarios, non-structural scenarios, and other scenarios. Zhang *et al.* [32] introduced GNNs on small graphs and giant graphs respectively. Quan *et al.* [33] and Zhang *et al.* [34] focused on reviewing works in a specific branch of GNNs, i.e. graph convolutional network (GCN). However, they seldom introduce GNNs works in traffic scenarios. Wu et.al proposed [35] the only survey spending a paragraph to describe GNNs in traffic domain, which is obviously not enough for anyone desiring to explore this field.

In summary, there still lacks a systematic and elaborated survey to explore the rapidly developed graph-based deep learning techniques in traffic domain recently. Our work aims to fill this gap and promote understanding of the new emerging techniques in transportation community.

III. Problems, Research Directions and Challenges

In this section, we introduce background knowledge in traffic domain briefly, including some important traffic problems and the corresponding research directions (as shown in Figure 1), as well as common challenges and techniques under these problems. On one hand, we believe that such a concise but systematic introduction can help readers understand this domain quickly. On the other hand, our survey shows that existing works related with graph-based deep learning techniques only cover some research directions, which inspires successors to transfer similar techniques to remaining directions.

A. Traffic Problems

The goals the transportation community aims to achieve include relieving traffic congestion, satisfying travel demand, enhancing traffic management, ensuring transportation safety and realizing automatic driving. Each problem under the corresponding traffic goal can be partitioned into several research directions and each direction can serve more than one problem.

1) Traffic Congestion: Traffic congestion [36] is one of the most important and urgent problems in modern cities in terms of significant time loss, air pollution and energy waste. The congestion can be solved by increasing the traffic efficiency [37], [38], alleviating the traffic congestion on road network [39]–[41], controlling the road conditions by traffic state prediction [42], [43], optimizing vehicle flow by controlling traffic signals [44], [45], optimizing passenger flow by predicting passenger demand in public transportation systems [46].

2) Travel Demand: The travel demand prediction refers to the demand of traffic services, such as taxi, bike, metro and bus in a crowd perspective. With the emerging of online ride-hailing platforms (e.g. Uber, DiDi) and rapid development of public transportation systems (e.g. metro system and bus system), travel demand prediction has become more and more important for transport authorities, business sectors and individuals. For related authorities, it can help to better allocate resources, e.g. increase metro frequency at rush hours, add more buses to service hotspots. For business sector, it enables them to better manage taxi-hiring [47], carpooling [48], bike-sharing services [49], [50], and maximize their revenues. For individuals, it encourages users to consider various forms of transportation to decrease their commuting time and improve travel experience.

3) Transportation Safety: Transportation safety is an indispensable part of public safety. Traffic accidents can not only cause damage to victims, vehicles and road infrastructures, but also lead to traffic congestion and reduce efficiency of road network. Therefore, monitoring the traffic accidents is essential to avoid property loss and save life. Many researchers focus on directions such as detecting traffic incidents [51], predicting traffic accidents from social media data [52], predicting the injury severity of traffic accidents [53], [54], predicting prevention of accidents [55]–[57].

4) Traffic Surveillance: Nowadays, surveillance cameras have been widely deployed in city roads, generating numerous images and videos [28]. Such development has enhanced traffic surveillance, which includes traffic law enforcement, automatic toll collection [58] and traffic monitoring systems. The research directions of traffic surveillance include license plate detection [59], automatic vehicle detection [60], pedestrian detection [61].

5) Autonomous Driving: Recently, automatic driving vehicle has become a hot spot of research in transportation domain. Many tasks are related with visual recognition. The research directions of autonomous driving include lane/vehicle detection [62], pedestrian detection [63], traffic sign detection [64] and human/vehicle trajectory prediction [65].

B. Research Directions

Our survey of graph-based deep learning in traffic domain shows that existing works focus mainly on traffic state prediction, travel demand prediction, trajectory prediction. A few works focus on vehicle behavior classification [66], optimal dynamic electronic toll collection (DETC) scheme [58], path availability [67], traffic signal control [68]. To our best knowledge, traffic incident detection and vehicle detection have not been explored based in a graph view yet.

1) Traffic State Prediction: Traffic state in literatures refers to traffic flow, traffic speed, travel time, traffic density and so on. Traffic Flow Prediction (TFP) [69], [70], Traffic Speed Prediction (TSP) [71], [72], Travel Time Prediction (TTP) [73]–[75] are hot branches of traffic state prediction and have attracted intensive studies.

2) Travel Demand Prediction: Travel demand prediction aims to estimate the future number of users who require traffic services. It can be categorized into two kinds, i.e. zone-level demand prediction and origin-destination travel demand prediction. The former one aims to predict the future travel demand in each region of a city, for example, to predict future taxi request in each area of a city [76], [77], or to predict the station-level passenger demand in subway system [46], [78]–[80] or to predict the bike hiring demand in each region of a city [49], [50]. The latter one aims to predict the number of travel demand from one region to another, which can provide richer information than the zone-level demand prediction and is a more challenging issue worth exploration. Up to now, there are only a few studies [81]–[83] focusing on the origin-destination based travel demand prediction, which is a promising research direction.

3) Traffic Signal Control: The traffic signal control aims to properly control the traffic lights so as to reduce vehicle staying time at the road intersections in the long run [26]. Traffic signal control [68] can optimize the traffic flow and reduce traffic congestion and vehicle emission.

4) Traffic Incident Detection: Major incidents can cause fatal injuries to travelers and long delays on a road network. Therefore, understanding the main cause of incidents and the impact of incidents on a traffic network is crucial for a modern transportation management system [51], [53], [54].

5) Human/Vehicle Trajectory Prediction: Trajectory Prediction [65], [84], [85] aims to forecast future positions of dynamic agents in a scene. Accurate human/vehicle trajectories prediction is of great importance for downstream tasks including autonomous driving and traffic surveillance [86]. For instance, an accurate pedestrian trajectory prediction can help controller to control the vehicle ahead in a dangerous environment [87]. It can also enable transportation surveillance system to identify suspicious activities [88].

C. Challenges and Techniques Overview

Although traffic problems and the related research directions are different, most of them share the same challenges, e.g. spatial dependency, temporal dependency, external factors.

1) Spatiotemporal Dependency: There are complex spatiotemporal dependency in traffic data which can affect the prediction in traffic tasks. For instance, to predict a traffic congestion in a region, its previous traffic conditions and the traffic conditions of its surrounding regions are important factors for prediction [36], [39], [40]. In vehicle trajectory prediction, the stochastic behaviors of surrounding vehicles and the historical information of self-trajectory influence the prediction performance [89]. When it comes to predict the ride-hailing demand in a region, its previous orders as well as orders in other regions with similar functionality are critical for prediction [90]. To predict the traffic signal, the geometric features of multiple intersections are taken into consideration, as well as the previous traffic flow around [68].

2) External Factors: Except the spatiotemporal data, some types of data play an important role in traffic tasks, referred as external factors, such as holidays, weather conditions (e.g. rainfall, temperature, air quality), extreme events [91] and traffic incidents (e.g. incident time, incident type) [92]. The influence of external factors on traffic conditions can be observed in daily life. A rainstorm is likely to affect the traffic volume. A large-scale concert or football match results in traffic congregation, affecting traffic conditions around.

To tackle challenges above, various deep learning techniques have been proposed. In this paper, we focus on graph-based deep learning architectures in traffic domain. Among these graph-based deep learning frameworks, graph neural networks (GNNs) are usually employed to model the spatial dependency in traffic network. Recurrent neural networks (RNNs) and temporal convolution network (TCN) are generally adopted to model the temporal dependency in traffic data. RNNs and Multi-layer Perceptrons (MLPs) are typically employed to process external factors. Sequence to Sequence (Seq2Seq) model is usually utilized to make multi-step traffic prediction. These techniques along with other tricks (e.g. gated mechanism, attention mechanism) are combined organically to improve the prediction accuracy.

In this paper, we aim to provide readers guidance about how to build a graph-based deep learning architecture and we have investigated enormous existing traffic works adopting graph-based deep learning solutions. In the following sections, we first introduce a common way to formulate the traffic problem and give detailed guidelines to build traffic graphs from various kinds of traffic data. Then we clarify the correlations between challenges and techniques (as shown in Figure 2) in two perspectives, i.e. the techniques perspective and the challenges perspective. In the perspective of techniques, we introduce several common techniques and interpret the way they tackle challenges in traffic tasks. In the perspective of challenges, we elaborate each challenge and summarize the techniques which can tackle this challenge. In a word, we hope to provide insights into solving traffic challenges with various deep learning techniques based on a graph view.

IV. Problem Formulation and Graph Construction

Among the graph-based deep learning traffic literatures we investigate, the majority of tasks (more than 80%) belong to

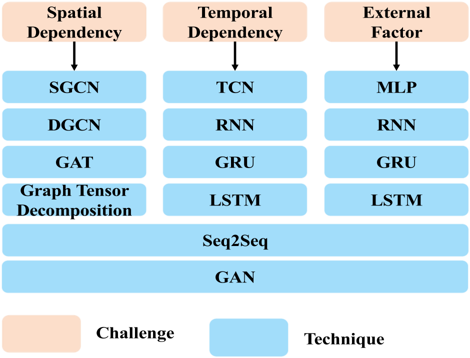


Fig. 2. Traffic challenges and the corresponding deep learning techniques. SGCN refers spectral graph convolution network, DGCN refers diffusion graph convolution network, GAT refers graph attention network, TCN refers temporal convolution network, RNN refers recurrent neural network, GRU refers gated recurrent unit, LSTM refers long short term memory network, MLP refers multi-layer perceptron, Seq2Seq refers sequence to sequence model, GAN refers generative adversarial network.

spatiotemporal forecasting problems, especially traffic state prediction and travel demand prediction. In this section, we first list commonly used notations. Then we summarize a general formulation of graph-based spatiotemporal prediction in traffic domain. We provide details to construct graphs from various traffic datasets. We also discuss multiple definitions of adjacency matrix, which represents the topology of graph-based traffic network and is the key element of graph-based solution.

A. Notations

In this section, we have denoted some commonly used notations, including graph related elements, variables, parameters (hyper or trainable), activation functions, and operations. The variables are comprised of input variables $\{x, X, \mathbf{x}, \mathbf{X}, \mathcal{X}\}$ and output variables $\{y, Y, \mathbf{y}, \mathbf{Y}, \mathcal{Y}\}$. These variables can divided into spatial variables, temporal variables, spatiotemporal variables. The spatial variables are only related with spatial attributes and the temporal variables are only related with temporal attributes. The spatiotemporal variables are related with both spatial and temporal attributes.

B. Graph-Based Spatio-Temporal Forecasting

To our best knowledge, most existing graph-based deep learning traffic works can be categorized into spatial-temporal forecasting due to that most traffic datasets have both spatial attributes and temporal attributes. They formalize their prediction problems in a very similar way despite different mathematical notations and representations. We summarize their works to provide a general formulation for graph-based spatial-temporal problems in traffic domain.

The traffic network is represented as a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{A})$, which can be weighted [73], [93], [69] or unweighted [67], [94], [95], directed [67], [96], [97] or undirected [70], [98], depending on specific tasks. $\mathbf{V}$ is a set of nodes and $|\mathbf{V}| = \mathbf{N}$ refers $\mathbf{N}$ nodes in the graph. Each node represents a traffic object, which can be a sensor [70], [71], [99], a road segment [93], [100], [101], a road intersection [69], [73], [96]. $\mathbf{E}$ is a set of edges referring the connectivity between nodes.

$\mathbf{A} = (\mathbf{a}_{ij})_{\mathbf{N}\times\mathbf{N}} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$ is the adjacency matrix containing the topology information of the traffic network, which is valuable for traffic prediction. The entry $\mathbf{a}_{ij}$ in matrix $\mathbf{A}$ represents the node proximity and is different in various applications. It can be a binary value 0 or 1 [70], [94], [95].

TABLE I
NOTATIONS IN THIS PAPER

Graph related elements	
$\mathbf{G}$	Graph
$\mathbf{E}$	Edges of graph $\mathbf{G}$
$\mathbf{V}$	Vertices of graph $\mathbf{G}$
$\mathbf{A} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$	Adjacency matrix of graph $\mathbf{G}$
$\mathbf{A}^T \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$	The transpose matrix of $\mathbf{A}$
$\tilde{\mathbf{A}} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$	Equal to $\mathbf{A} + \mathbf{I}_\mathbf{N}$, a self-looped $\mathbf{A}$
$\mathbf{D} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$	The degree matrix of adjacency matrix $\mathbf{A}$
$\mathbf{D}_\mathbf{I} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$	The in-degree matrix of adjacency matrix $\mathbf{A}$
$\mathbf{D}_\mathbf{O} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$	The out-degree matrix of adjacency matrix $\mathbf{A}$
$\mathbf{L} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$	Laplacian matrix of graph $\mathbf{G}$
$\mathbf{U} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$	The eigenvectors matrix of $\mathbf{L}$
$\mathbf{\Lambda} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$	The diagonal eigenvalues matrix of $\mathbf{L}$
$\boldsymbol{\lambda}_{max}$	The max eigenvalue of $\mathbf{L}$
$\mathbf{I}_\mathbf{N} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$	An identity matrix
Hyper parameters	
$\mathbf{N}$	The number of nodes in graph $\mathbf{G}$
$\mathbf{F}_\mathbf{I}$	The number of input features
$\mathbf{F}_\mathbf{H}$	The number of hidden features
$\mathbf{F}_\mathbf{O}$	The number of output features
$\mathbf{P}$	The number of past time slices
$\mathbf{Q}$	The number of future time slices
$\mathbf{d}$	The dilation rate
Trainable parameters	
W, b, θ, ϕ	The trainable parameters
Θ	The kernel
Activation functions	
$\boldsymbol{\rho}(\cdot)$	The activation function, e.g. tanh, sigmoid, ReLU
$\boldsymbol{\sigma}(\cdot) \in [0, 1]$	The sigmoid function
$\boldsymbol{tanh}(\cdot) \in [-1, 1]$	The hyperbolic tangent function
$\boldsymbol{ReLU}(\cdot) \in [0, x]$	The ReLU function
Operations	
$*_\mathcal{G}$	The convolution operator on graph
$\odot$	Element-wise multiplication
$\cdot$	Matrix multiplication
Spatial variables	
$X \in \mathbb{R}^{\mathbf{N}\times\mathbf{F}_\mathbf{I}}$	An input graph composed of $\mathbf{N}$ nodes with $\mathbf{F}_\mathbf{I}$ features
$X_j \in \mathbb{R}^{\mathbf{N}}$	The j^{th} feature of an input graph
$X^i \in \mathbb{R}^{\mathbf{F}_\mathbf{I}}$	Node i in an input graph
$x \in \mathbb{R}^{\mathbf{N}}$	A simply input graph
$Y \in \mathbb{R}^{\mathbf{N}\times\mathbf{F}_\mathbf{O}}$	An output graph composed of $\mathbf{N}$ nodes with $\mathbf{F}_\mathbf{O}$ features
$Y_j \in \mathbb{R}^{\mathbf{N}}$	The j^{th} feature of an output graph
$Y^i \in \mathbb{R}^{\mathbf{F}_\mathbf{O}}$	Node i in an output graph
$y \in \mathbb{R}^{\mathbf{N}}$	A simply output graph
Temporal variables	
$\mathbf{X} \in \mathbb{R}^{\mathbf{P}\times\mathbf{F}_\mathbf{I}}$	A sequential input with $\mathbf{F}_\mathbf{I}$ features over $\mathbf{P}$ time slices
$\mathbf{X}_t \in \mathbb{R}^{\mathbf{F}_\mathbf{I}}$	The element of sequential input at time t
$\mathbf{x} \in \mathbb{R}^{\mathbf{P}}$	A simply sequential input over $\mathbf{P}$ time slices
$\mathbf{x}_t \in \mathbb{R}$	The element of simply sequential input at time t
$\mathbf{H}_t \in \mathbb{R}^{\mathbf{F}_\mathbf{H}}$	A hidden state with $\mathbf{F}_\mathbf{H}$ features at time t
$\mathbf{Y} \in \mathbb{R}^{\mathbf{P}\times\mathbf{F}_\mathbf{O}}$	A sequential output with $\mathbf{F}_\mathbf{O}$ features over $\mathbf{P}$ time slices
$\mathbf{Y}_t \in \mathbb{R}^{\mathbf{F}_\mathbf{O}}$	The element of sequential output at time t
$\mathbf{y} \in \mathbb{R}^{\mathbf{P}}$	A simply sequential output over $\mathbf{P}$ time slices
$\mathbf{y}_t \in \mathbb{R}$	The element of simply sequential output at time t
Spatiotemporal variables	
$\mathcal{X} \in \mathbb{R}^{\mathbf{P}\times\mathbf{N}\times\mathbf{F}_\mathbf{I}}$	A series of input graphs composed of $\mathbf{N}$ nodes with $\mathbf{F}_\mathbf{I}$ features over $\mathbf{P}$ time slices
$\mathcal{X}_t \in \mathbb{R}^{\mathbf{N}\times\mathbf{F}_\mathbf{I}}$	An input graph at time t
$\mathcal{X}_t^i \in \mathbb{R}^{\mathbf{F}_\mathbf{I}}$	node i in an input graph at time t
$\mathcal{X}_{t,j} \in \mathbb{R}^{\mathbf{N}}$	the j^{th} feature of an input graph at time t
$\mathcal{X}_{t,j}^i \in \mathbb{R}$	the j^{th} feature of node i in an input graph at time t
$\mathcal{Y} \in \mathbb{R}^{\mathbf{P}\times\mathbf{N}\times\mathbf{F}_\mathbf{O}}$	A series of output graphs composed of $\mathbf{N}$ nodes with $\mathbf{F}_\mathbf{O}$ features over $\mathbf{P}$ time slices
$\mathcal{Y}_t \in \mathbb{R}^{\mathbf{N}\times\mathbf{F}_\mathbf{O}}$	An output graph at time t
$\mathcal{Y}_t^i \in \mathbb{R}^{\mathbf{F}_\mathbf{O}}$	node i in an output graph at time t
$\mathcal{Y}_{t,j} \in \mathbb{R}^{\mathbf{N}}$	the j^{th} feature of an output graph at time t
$\mathcal{Y}_{t,j}^i \in \mathbb{R}$	the j^{th} feature of node i in an output graph at time t

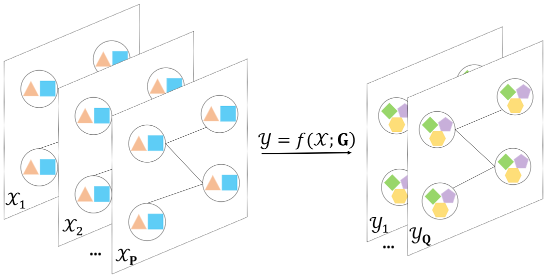


Fig. 3. The graph-based spatial-temporal problem formulation in traffic domain.

Specifically, 0 indicates no edge between node i and node j while 1 indicates an edge between these two nodes. It can also be a float value representing some kind of relationship between nodes [93], [102], e.g. the road distance between two sensors [71], [103], [97].

$\mathcal{X}_t = [\mathcal{X}_t^1, \cdots, \mathcal{X}_t^i, \cdots, \mathcal{X}_t^\mathbf{N}] \in \mathbb{R}^{\mathbf{N}\times\mathbf{F_I}}$ is a feature matrix of the whole graph at time t. $\mathcal{X}_t^i \in \mathbb{R}^{\mathbf{F_I}}$ represents node i with $\mathbf{F_I}$ features at time t. The features are usually traffic indicators, such as traffic flow [97], [98], traffic speed [71], [100], [96], or rail-hail orders [90], [102], passenger flow [78], [79]. Usually, continuous indicators are normalized during data preprocessing phase.

Given historical indicators of the whole traffic network over past $\mathbf{P}$ time slices, denoted as $\mathcal{X} = [\mathcal{X}_1, \cdots, \mathcal{X}_t, \cdots, \mathcal{X}_\mathbf{P}] \in \mathbb{R}^{\mathbf{P}\times\mathbf{N}\times\mathbf{F_I}}$, the spatial-temporal forecasting problem in traffic domain aims to predict the future traffic indicators over the next $\mathbf{Q}$ time slices, denoted as $\mathcal{Y} = [\mathcal{Y}_1, \cdots, \mathcal{Y}_t, \cdots, \mathcal{Y}_\mathbf{Q}] \in \mathbb{R}^{\mathbf{Q}\times\mathbf{N}\times\mathbf{F_O}}$, where $\mathcal{Y}_t \in \mathbb{R}^{\mathbf{N}\times\mathbf{F_O}}$ represents output graph with $\mathbf{F_O}$ features at time t. The problem (as shown in Figure 3) can be formulated as follows:

$$\mathcal{Y} = f(\mathcal{X}; \mathbf{G}) \tag{1}$$

Some works predict multiple traffic indicators in the future (i.e. $\mathbf{F_O} > 1$) while other works predict one traffic indicator (i.e. $\mathbf{F_O} = 1$), such as traffic speed [96], [100], rail-hide orders [90], [102]. Some works only consider one-step prediction [76], [104], [58], i.e. forecasting traffic conditions in the next time step and $\mathbf{Q} = 1$. But models designed for one-step prediction can't be directly applied to predict multiple steps, because they are optimized by reducing error during the training stage for the next-step instead of the subsequent time steps [77]. Many works focus on multi-step forecasting (i.e. $\mathbf{Q} > 1$) [43], [105], [106]. According to our survey, there are mainly three kinds of techniques to generate a multi-step output, i.e. FC layer, Seq2Seq, dilation technique. Fully connected (FC) layer is the simplest technique as being the output layer to obtain a desired output shape [70], [71], [94], [107], [92], [108]. Some works adopt the Sequence to Sequence (Seq2Seq) architecture with a RNNs-based decoder to generate output recursively through multiple steps [99], [109], [105], [110], [97], [111]. Dilation technique is adopted to get a desired output length [103], [106]. In addition, some works not only consider traffic indicators, but also take external factors (e.g. time attributes, weather) [71], [113], [92], [114] into consideration. Therefore, the problem formulation becomes:

$$\mathcal{Y} = f(\mathcal{X}, \mathcal{E}; \mathbf{G}) \tag{2}$$

where $\mathcal{E}$ is the external factors.

C. Graph Construction From Traffic Datasets

To model a traffic network as a graph is vital for any works that intend to utilize graph-based deep learning architectures to solve traffic problems. A traffic graph $\mathbf{G}$ for prediction is generally composed of four parts, i.e. nodes $\mathbf{V}$, node features (feature matrix $\mathcal{X}_t$), edges $\mathbf{E}$, edge weight $\mathbf{a}_{ij}$. Note that edges and edge weight can be represented by adjacency matrix $\mathbf{A} = (\mathbf{a}_{ij})_{\mathbf{N}\times\mathbf{N}}$. Nodes and node features can be constructed from traffic datasets. The construction of adjacency matrix not only depends on traffic datasets but also depends on the assumption of node relationship, which can be static or dynamic. We first introduce how to construct node and node features from various kinds of traffic datasets and then we give a systematic introduction to the popular adjacency matrices.

1) Nodes and Node Features Construction: Many works are different in graph construction due to the different traffic datasets they collect. We divide these datasets into four categories according to the traffic infrastructures: data collected by the sensors deployed on road network [70], [71], [72], vehicle GPS trajectories [69], [112], [96], orders of rail-hailing system [77], [102], [114], transaction records of subway system [78], [79] or bus system [112]. For each category, we describe the datasets and explain the construction of nodes $\mathbf{V}$, feature matrix $\mathcal{X}_t$.

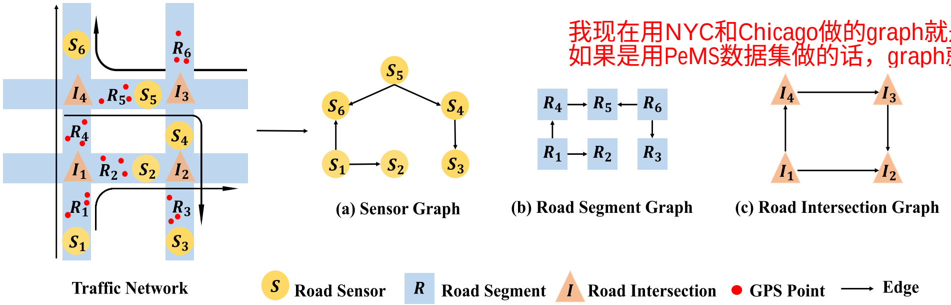
Sensors Datasets: Traffic measurements (e.g. traffic speed) are generally collected during a short time interval by the sensors (e.g. loop detectors, probes) on a road network in metropolises like Beijing [93], California [72], Los Angeles [71], New York [100], Philadelphia [107], Seattle [95], Xiamen [99], and Washington [107]. Sensor datasets are the most prevalent datasets in existing works, especially PEMS dataset from California. Generally, a road network contains traffic objects such as sensors, road segments.

A sensor graph (as shown in Figure 4) is constructed in [70], [71], [97] where a sensor represents a node and features of this node are traffic measurements collected by its corresponding sensor.

A road segment graph (as shown in Figure 4) is constructed in [93], [100], [107] where a road segment represents a node and features of this node are average traffic measurements (e.g. traffic speed) recorded by all the sensors on its corresponding road segment.

GPS Datasets: GPS trajectories datasets are usually generated by numbers of taxis over some period of time in a city, e.g. Beijing [69], Chengdu [69], Shenzhen [94], Cologne [96], and Chicago [101]. Each taxi produces substantial GPS records with time, location, speed information every day. Every GPS record is fitted to its nearest road on the city road map. All roads are divided into multiple road segments through road intersections.

A road segment graph (as shown in Figure 4) is constructed in [94], [101] where a road segment represents a node and



我现在用NYC和Chicago做的graph就是(b)road segment graph！
如果是用PeMS数据集做的话，graph就是(a)road sensor!

Fig. 4. Graph construction from various traffic datasets: a) In a sensor graph, sensor represents node and there is an edge between adjacent sensors on the same side of a road. b) In a road segment graph, road segment represents node and two connected segments have an edge. c) In a road intersection graph, road intersection represents node and there is an edge between two road intersections connected by a road segment. Most works consider the edge direction being the traffic flow direction [71], [99], [67], [97], [69], [112], while some works ignore the direction and construct an undirected graph [70], [103], [95], [96], [101].

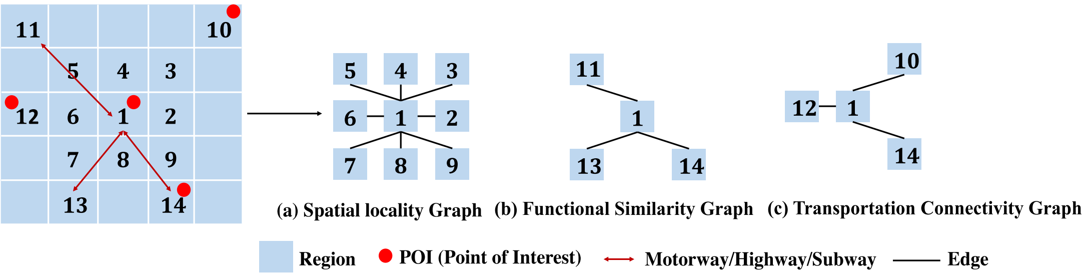

(a) Spatial locality Graph
(b) Functional Similarity Graph
(c) Transportation Connectivity Graph
Region
POI (Point of Interest)
Motorway/Highway/Subway
Edge

Fig. 5. Multi-relationships: a) A spatial locality graph: This graph is based on spatial proximity and it constructs edges between a region and its 8 adjacent regions in a 3 x 3 grid. b) A functional similarity graph: This graph assumes that regions sharing similar functionality might have similar demand patterns. Edges are constructed between regions with similar surrounding POIs (Point of Interests). c) A transportation connectivity graph: This graph assumes that regions which are geographically distant from the target region but conveniently reachable by transportation (e.g. motorway, highway or subway) have strong correlations with the target region. There should be edges between them.

features of this node are average traffic measurements recorded by all the GPS points on its corresponding road segment.

A road intersection graph (as shown in Figure 4) is constructed in [69], [73], [96] where a road intersection represents a node and features of this node are sum-up of the traffic measurements through it.

Rail-Hailing Datasets: These datasets record car/taxi/bicycle demand orders over a period of time in cities like Beijing [90], [102], Chengdu [102], and Shanghai [90], Manhattan, New York [100]. The target city with an OpenStreetMap is divided into equal-size grid-based regions (as shown in Figure 5). Each region is defined as a node in a graph. The feature of each node is the number of orders in the corresponding region during a given interval.

Transactions Datasets: These datasets are collected by automatic fare collection (AFC) system deployed in public transit network, such as subway network and bus network. A subway graph is constructed in [78], [79], [112]. Each station in the subway system is treated as a node. The features of a station usually contain the number of passengers departing at the station and the number of passengers arriving at the station during a given time interval based on transaction records collected by subwayAFC systems, which log when each passenger enters and leaves a metro system.

A bus graph is constructed in [112]. Each bus stop is treated as a node. The features of a bus stop usually contain the number of departing passengers at the station during a given time interval, but not the number of arriving passengers, since most bus AFC systems only log the boarding record of each passenger.

2) Adjacency Matrix Construction: The adjacency matrix $\mathbf{A} = (\mathbf{a}_{ij})_{\mathbf{N}\times\mathbf{N}} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$ is the key to capture spatial dependency which is valuable for prediction. Element $\mathbf{a}_{ij}$ (unweighted or weighted) represents heterogeneous pairwise relationship between nodes. However, there are different assumptions of node relationships in different traffic scenarios, based on which the adjacency matrix can be designed differently, e.g. fixed matrix, dynamic matrix, evolving matrix.

Fixed Matrix: Many works assume that the correlations between nodes are fixed and do not change over time. Therefore, a fixed matrix is designed and unchanged during the whole experiment. Researchers have designed various fixed adjacency matrices to capture various kinds of pre-defined correlations between nodes in a traffic graph, like function similarity and transportation connectivity [90], semantic connection [102], temporal similarity [72]. Here, we introduce several popular adjacency matrices.

Connection matrix measures the connectivity between nodes. The entry value in the matrix is defined as 1 (connection) or 0 (disconnection) [70], [107], [94], [95].

Distance matrix measures the closeness between nodes in terms of geometrical distance. The entry value is defined as a function of distance between nodes [86]. For example, some works [69], [73], [98], [101], [77], [96] used threshold

My paper的Adjacency Matrix要搞3个出来；
第一个就是简单的描述node与node连接的connection matrix (非1即0)；
第二个参考Traffic RisK文章,建立一个A_D;
第三个参考Traffic RisK文章,建立一个A_P.

Gaussian Kernel to define $\mathbf{a}_{ij}$ as follows:

$$\mathbf{a}_{ij} = \begin{cases} \exp\left(-\frac{\mathbf{d}_{ij}^2}{\sigma^2}\right), & i \neq j \text{ and } \mathbf{d}_{ij} \geq \epsilon \\ 0, & i = j \text{ or } \mathbf{d}_{ij} < \epsilon \end{cases} \quad (3)$$

where $\mathbf{d}_{ij}$ is the distance between node i and node j. Hyper parameters σ^2 and ϵ are thresholds to control the distribution and sparsity of matrix $\mathbf{A}$.

Functional similarity matrix measures whether two nodes are similar in terms of functionality (e.g. both of them are business zones). The corresponding functional similarity graph is shown in Figure 5. It assumes that regions sharing similar functionality might have similar demand patterns [90]. Edges are constructed between regions with similar surrounding POIs (Point of Interests).

Transportation connectivity matrix measures the correlation between regions that are geographically distant but conveniently reachable by motorway, highway or subway. The corresponding transportation connectivity graph is shown in Figure 5. There should be edges between them [90].

Dynamic Matrix: Some works argue that the pre-defined matrix does not necessarily reflect the true dependency among nodes due to the defective prior knowledge or incomplete data [73]. A novel adaptive matrix is proposed and learned through data. Experiments in [73], [103], [100] have proven that adaptive matrix can precisely capture the hidden spatial dependency more precisely in some traffic tasks.

Evolving Matrix: In some scenarios, the graph structure can evolve over time as some edges may become unavailable, like road congestion or closure, and become available again after alleviating congestion. An evolving topological structure [67], [115] is incorporated into the model to capture such dynamic spatial change.

V. Deep Learning Techniques Perspective

We summarize the graph-based deep learning architectures in existing traffic literatures and find that most of them are composed of graph neural networks (GNNs) and other modules, such as recurrent neural networks (RNNs), temporal convolution network (TCN), Sequence to Sequence (Seq2Seq) model, generative adversarial network (GAN) (as shown in Table II). It is the cooperation of GNNs and other deep learning techniques that achieves state-of-the-art performance in many traffic scenarios. This section aims to introduce the functionality, advantages, defects and variants of these techniques in traffic tasks, which can help participators understand how to utilize these deep learning techniques in traffic domain.

A. GNNs

In the last couple of years, motivated by the huge success of deep learning approaches (e.g. CNNs, RNNs), there is an increasing interest in generalizing neural networks to arbitrarily structured graphs and such networks are classified as graph neural networks (GNNs). In the early stage, the studies about GNNs can be categorized into recurrent graph neural networks (RecGNNs) which are inspired by RNNs [35]. Subsequently, inspired by the huge success of CNNs, many works focus on extending the convolution of CNN on graph data and these works can be categorized into convolutional graph neural networks (ConvGNNs) [35]. There are also other branches of GNNs developed in recent years, e.g. graph auto-encoders (GAEs) [122] and graph attention networks (GATs) [123]. According to our investigation, most traffic works focus on ConvGNNs and there are only a few studies [120] employing other branches of GNNs up to now. Further, ConvGNNs can be divided into two main streams, i.e. the spectral-based approaches which develop graph convolutions based on the spectral theory and the spatial-based approaches which define graph convolutions based on spatial relations between nodes [124]. Recently, many novel spatial-based convolutions have emerged, among which diffusion convolution is a popular spatial-based graph convolution which regards graph convolution as a diffusion process.

According to our survey, most existing traffic works utilize either spectral graph convolution or diffusion graph convolution. There are also other novel convolutions [69] but their applications in traffic domain are relatively few. Therefore, in this section, we focus on introducing spectral graph convolution (SGC) and diffusion graph convolution (DGC) in traffic domain. In this paper, we refer the graph neural network with spectral graph convolution as SGCN and that with diffusion graph convolution as DGCN. Note that SGC is for undirected graph while DGC can be applied in both directed graph and undirected graph. In addition, both SGC and DGC aim to generate new feature representations for each node in a graph through feature aggregation and non-linear transformation (as shown in Figure 6).

1) Spectral Graph Convolution: In the spectral theory, a graph is represented by its corresponding normalized Laplacian matrix $\mathbf{L} = \mathbf{I_N} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$. The real symmetric matrix $\mathbf{L}$ can be diagonalized via eigendecomposition as $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^T$ where $\mathbf{U} \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$ is the eigenvectors matrix and $\Lambda \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$ is the diagonal eigenvalues matrix. Since $\mathbf{U}$ is also an orthogonal matrix, Shuman *et al.* [125] adopted it as a graph Fourier basis, defining graph Fourier transform of a graph signal $x \in \mathbb{R}^{\mathbf{N}}$ as $\hat{x} = \mathbf{U}^T x$, and its inverse as $x = \mathbf{U}\hat{x}$.

Bruna *et al.* [126] tried to build an analogue of CNN convolution in spectral domain and defined the spectral convolution as $y = \Theta *_{\mathcal{G}} x = \mathbf{U}\Theta\mathbf{U}^T x$, i.e. transforming x into spectral domain, adjusting its amplitude by a diagonal kernel $\Theta = \text{diag}(\theta_0, \ldots, \theta_{\mathbf{N}-1}) \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$, and doing inverse Fourier transform to get the final result y in spatial domain. Although such convolution is theoretically guaranteed, it is computationally expensive as multiplication with $\mathbf{U}$ is $\mathcal{O}(\mathbf{N}^2)$ and the eigendecomposition of $\mathbf{L}$ is intolerable for large scale graphs. In addition, it considers all nodes by the kernel Θ with $\mathbf{N}$ parameters and can't extract spatial localization.

To avoid such limitations, Defferrard *et al.* [127] localized the convolution and reduced its parameters by restricting the kernel Θ to be a polynomial of eigenvalues matrix Λ as $\Theta = \sum_{k=0}^{\mathbf{K}-1} \theta_k \Lambda^k$ and $\mathbf{K}$ determines the maximum radius of the convolution from a central node. Thus, the convolution can be rewritten as $\Theta *_{\mathcal{G}} x = \sum_{k=0}^{\mathbf{K}-1} \theta_k \mathbf{U}\Lambda^k\mathbf{U}^T x = \sum_{k=0}^{\mathbf{K}-1} \theta_k \mathbf{L}^k x$.

TABLE II

THE DECOMPOSITION OF GRAPH-BASED DEEP LEARNING ARCHITECTURES INVESTIGATED IN THIS PAPER

Reference	**Year**	**Directions**	**Models**	**Modules**
[65]	2019	Human/Vehicle Trajectory Prediction	SAGCN	SGCN, TCN, Attention
[88]	2019	Human/Vehicle Trajectory Prediction	Social-BiGAT	GAT, LSTM, GAN
[86]	2020	Human/Vehicle Trajectory Prediction	Social-STGCNN	SGCN, TCN
[84]	2020	Human/Vehicle Trajectory Prediction	Social-WaGDAT	GAT, Seq2Seq, MLP
[89]	2020	Human/Vehicle Trajectory Prediction		SGCN, LSTM
[58]	2019	Optimal DETC Scheme		SGCN
[66]	2020	Vehicle Behaviour Classification	MR-GCN	SGCN, LSTM
[68]	2018	Traffic Signal Control		SGCN, Reinforcement Learning
[67]	2019	Path Availability	LRGCN-SAPE	SGCN, LSTM
[73]	2019	Travel Time Prediction		SGCN
[69]	2018	Traffic Flow Prediction	KW-GCN	SGCN, LCN
[98]	2018	Traffic Flow Prediction	Graph-CNN	CNN, Graph Matrix
[116]	2018	Traffic Flow Prediction	DST-GCNN	SGCN
[70]	2019	Traffic Flow Prediction		SGCN, CNN, Attention Mechanism
[112]	2019	Traffic Flow Prediction		SGCN, TCN, Residual
[110]	2019	Traffic Flow Prediction	GHCRNN	SGCN, GRU, Seq2Seq
[105]	2019	Traffic Flow Prediction	STGSA	GAT, GRU, Seq2Seq
[97]	2019	Traffic Flow Prediction	DCRNN-RIL	DGCN, GRU, Seq2Seq
[117]	2019	Traffic Flow Prediction	MVGCN	SGCN, FNN, Gate Mechanism, Residual
[118]	2019	Traffic Flow Prediction	STGI- ResNet	SGCN, Residual
[119]	2020	Traffic Flow Prediction	FlowConvGRU	DGCN, GRU
[46]	2020	Traffic Flow Prediction	Multi-STGCnet	SGCN, LSTM
[120]	2018	Traffic Speed Prediction		GAT, GRU, Gate Mechanism
[71]	2019	Traffic Speed Prediction	GTCN	SGCN, TCN, Residual
[72]	2019	Traffic Speed Prediction	3D-TGCN	SGCN, Gate Mechanism
[92]	2019	Traffic Speed Prediction	DIGC-Net	SGCN, LSTM
[121]	2019	Traffic Speed Prediction	MW-TGC	SGCN, LSTM
[111]	2019	Traffic Speed Prediction	AGC-Seq2Seq	SGCN, GRU, Seq2Seq, Attention Mechanism
[96]	2019	Traffic Speed Prediction	GCGA	SGCN, GAN
[108]	2019	Traffic Speed Prediction	ST-GAT	GAT, LSTM
[93]	2018	Traffic State Prediction	STGCN	SGCN, TCN, Gate Mechanism
[109]	2018	Traffic State Prediction	DCRNN	DGCN, GRU, Seq2Seq
[100]	2019	Traffic State Prediction		SGCN, CNN, Gate Mechanism
[113]	2019	Traffic State Prediction	MRes-RGNN	DGCN, GRU, Residual, Gate Mechanism
[101]	2019	Traffic State Prediction	GCGAN	DGCN, LSTM, GAN, Seq2Seq, Attention Mechanism
[103]	2019	Traffic State Prediction	Graph WaveNet	DGCN, TCN, Residual, Gate Mechanism
[94]	2019	Traffic State Prediction	T-GCN	SGCN, GRU
[95]	2019	Traffic State Prediction	TGC-LSTM	SGCN, LSTM
[42]	2019	Traffic State Prediction	DualGraph	Seq2Seq, MLP, Graph Matirx
[106]	2019	Traffic State Prediction	ST-UNet	SGCN, GRU
[99]	2020	Traffic State Prediction	GMAN	GAT, Gate Mechanism, Seq2Seq, Attention Mechanism
[107]	2020	Traffic State Prediction	OGCRNN	SGCN, GRU, Attention Mechanism
[43]	2020	Traffic State Prediction	MRA-BGCN	SGCN, GRU, Seq2Seq, Attention Mechanism
[49]	2018	Travel Demand-Bike		SGCN, LSTM, Seq2Seq
[50]	2018	Travel Demand-Bike	GCNN-DDGF	SGCN, LSTM
[78]	2020	Travel Demand-Subway	PVCGN	SGCN, GRU, Seq2Seq, Attention Mechanism
[79]	2019	Travel Demand-Subway	WDGTC	Tensor Completion, Graph Matrix
[90]	2019	Travel Demand-Taxi	CGRNN	SGCN, RNN, Attention Mechanism, Gate Mechanism
[102]	2019	Travel Demand-Taxi	GEML	SGCN, LSTM
[76]	2019	Travel Demand-Taxi	MGCN	SGCN
[77]	2019	Travel Demand-Taxi	STG2Seq	SGCN, Seq2Seq, Attention Mechanism, Gate Mechanism, Residual
[114]	2019	Travel Demand-Taxi		SGCN, LSTM, Seq2Seq
[104]	2019	Travel Demand-Taxi	ST-ED-RMGC	SGCN, LSTM, Seq2Seq, Residual

Further more, Defferrard *et al.* [127] adopted the Chebyshev polynomials $T_k(x)$ to approximate $\mathbf{L}^k$, resulting in $\Theta \ast_{\mathcal{G}} x \approx \sum_{k=0}^{\mathbf{K}-1} \theta_k T_k(\tilde{\mathbf{L}})x$ with a rescaled $\tilde{\mathbf{L}} = \frac{2}{\boldsymbol{\lambda}_{\max}}\mathbf{L}-\mathbf{I_N}$ where $\boldsymbol{\lambda}_{\max}$ is the largest eigenvalue of $\mathbf{L}$ and $T_k(x) = 2xT_{k-1}(x)-T_{k-2}(x)$, $T_0(x) = 1$, $T_1(x) = x$ [128]. By recursively computing $T_k(x)$, the complexity of this $\mathbf{K}$-localized convolution can be reduced to $\mathcal{O}(\mathbf{K}|\mathbf{E}|)$ with $|\mathbf{E}|$ being the number of edges.

Based on [127], Kipf *et al.* [129] simplified the spectral graph convolution by limiting $\mathbf{K} = 2$ and with $T_0(\tilde{\mathbf{L}}) = 1$, $T_1(\tilde{\mathbf{L}}) = \tilde{\mathbf{L}}$. They got $\Theta \ast_{\mathcal{G}} x \approx \theta_0 T_0(\tilde{\mathbf{L}})x + \theta_1 T_1(\tilde{\mathbf{L}})x = \theta_0 x + \theta_1 \tilde{\mathbf{L}}x$. Noticing that $\tilde{\mathbf{L}} = \frac{2}{\lambda_{\max}}\mathbf{L}-\mathbf{I_N}$, they set $\boldsymbol{\lambda}_{\max} = 2$, resulting in $\Theta \ast_{\mathcal{G}} x \approx \theta_0 x + \theta_1(\mathbf{L} - \mathbf{I_N})x$. For that $\mathbf{L} = \mathbf{I_N}-\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ and $\mathbf{L}-\mathbf{I_N} = -\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, they got $\Theta \ast_{\mathcal{G}} x \approx \theta_0 x - \theta_1(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})x$. Further, they reduced the number of parameters by setting $\theta = \theta_0 = -\theta_1$ to address overfitting and got $\Theta \ast_{\mathcal{G}} x \approx \theta(\mathbf{I_N} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})x$. They defined $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I_N}$ and adopted a renormalization trick to get $y = \Theta \ast_{\mathcal{G}} x \approx \theta\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}x$, where $\tilde{\mathbf{D}}$ is the degree matrix of $\tilde{\mathbf{A}}$. Finally, Kipf *et al.* [129] proposed a spectral graph convolution layer as follows:

$$
\begin{aligned}
Y_j &= \boldsymbol{\rho}(\Theta_j \ast_{\mathcal{G}} X) = \boldsymbol{\rho}(\sum_{i=1}^{\mathbf{F_I}} \theta_{i,j}\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}X_i),\ 1 \le j \le \mathbf{F_O} \\
Y &= \boldsymbol{\rho}(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}XW) \qquad (4)
\end{aligned}
$$

here, $X \in \mathbb{R}^{\mathbf{N}\times\mathbf{F_I}}$ is the layer input with $\mathbf{F_I}$ features, $X_i \in \mathbb{R}^{\mathbf{N}}$ is its i^{th} feature. $Y \in \mathbb{R}^{\mathbf{N}\times\mathbf{F_O}}$ is the layer output with $\mathbf{F_O}$ features, $Y_j \in \mathbb{R}^{\mathbf{N}}$ is its j^{th} feature. $W \in \mathbb{R}^{\mathbf{F_I}\times\mathbf{F_O}}$ is a trainable parameter. $\boldsymbol{\rho}(\cdot)$ is the activation function. Such layer can aggregate information of 1-hop neighbors. The receptive

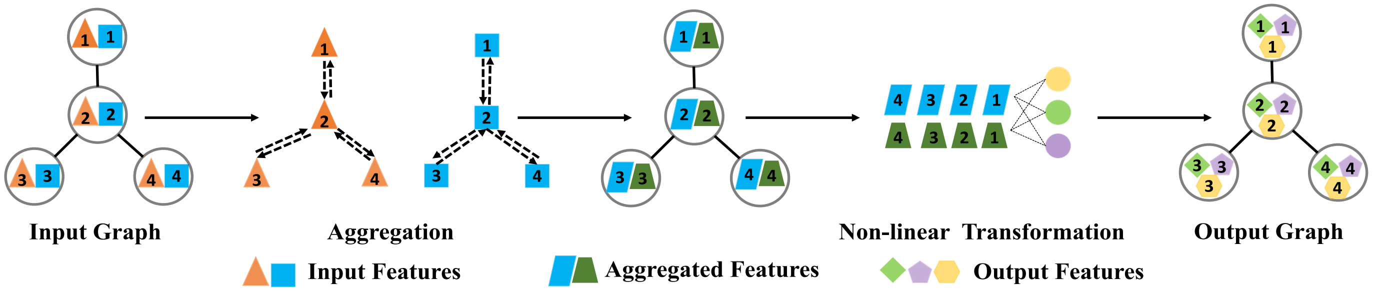


Fig. 6. The structure of Graph Neural Network is generally composed of two kind of layers: 1) Aggregation layer: In each feature dimension, the features of adjacent nodes are aggregated to the central node. Mathematically, the output of aggregation layer is the product of adjacency matrix and features matrix. 2) Non-linear transformation layer: All the aggregated features of each node are fed into the non-linear transformation layer to create higher level feature representation. All nodes share the same transformation kernel. $\{1, 2, 3, 4\}$ are node indexes.

field of neighborhood can be expanded by stacking multiple graph convolution layers [43].

2) Diffusion Graph Convolution: Spectral graph convolution requires a symmetric Laplacian matrix to implement eigendecomposition. It becomes invalid for a directed graph with an asymmetric Laplacian matrix. Diffusion convolution origins from graph diffusion and has no constraint on graph. Graph diffusion [130] can be represented as a transition matrix power series giving the probability of jumping from one node to another node at each step. After many steps, such Markov process converges to a stationary distribution $\mathcal{P} = \sum_{k=0}^{\infty} \alpha(1-\alpha)^k (\mathbf{D_O}^{-1}\mathbf{A})^k$, where $\mathbf{D_O}^{-1}\mathbf{A}$ is the transition matrix, $\alpha \in [0, 1]$ is the restart probability and k is the diffusion step. In practice, a finite $\mathbf{K}$-step truncation of the diffusion process is adopted and each step is assigned a trainable weight θ. Based on the $\mathbf{K}$-step diffusion process, Li *et al.* [109] defined diffusion graph convolution as:

$$y = \Theta *_{\mathcal{G}} x = \sum_{k=0}^{\mathbf{K}-1} (\theta_{k,1}(\mathbf{D_O}^{-1}\mathbf{A})^k + \theta_{k,2}(\mathbf{D_I}^{-1}\mathbf{A}^T)^k)x \quad (5)$$

here, $\mathbf{D_O}^{-1}\mathbf{A}$ represents the transition matrix and $\mathbf{D_I}^{-1}\mathbf{A}^T$ is its transpose. Such bidirectional diffusion enables the operation to capture the spatial correlation on a directed graph [109]. Similar to spectral graph convolution layer, a diffusion graph convolutional layer is built as follows:

$$Y_j = \rho(\sum_{k=0}^{\mathbf{K}-1} \sum_{i=1}^{\mathbf{F_I}} (\theta_{k,1,i,j}(\mathbf{D_O}^{-1}\mathbf{A})^k + \theta_{k,2,i,j}(\mathbf{D_I}^{-1}\mathbf{A}^T)^k)X_i)$$
$$Y = \rho(\sum_{k=0}^{\mathbf{K}-1} (\mathbf{D_O}^{-1}\mathbf{A})^k X W_{k1} + (\mathbf{D_I}^{-1}\mathbf{A}^T)^k X W_{k2}) \quad (6)$$

where $1 \leq j \leq \mathbf{F_O}$, parameters $W_{k1}, W_{k2} \in \mathbb{R}^{\mathbf{F_I} \times \mathbf{F_O}}$ are trainable.

3) GNNs in Traffic Domain: Many traffic works, such as subway network and road network, are graph structure naturally (See Section IV). Compared with previous works modeling traffic network as grids [131], [132], the works modeling traffic network as graph can fully utilize spatial information.

By now, many works employ convolution operation directly on traffic graph to capture the complex spatial dependency of traffic data. Most of them adopt spectral graph convolution (SGC) while some employ diffusion graph convolution (DGC) [101], [103], [109], [113], [97], [119]. There are also some other graph based deep learning techniques such as graph attention network (GAT) [99], [105], [108], [120], tensor decomposition and completion on graph [79], but their related works are few, which might be a future research direction.

The key difference between SGC and DGC lies in their matrices which represent different assumptions on the spatial correlations in traffic network. The adjacency matrix in SGC infers that a central node in a graph has stronger correlation with its adjacent nodes than other distant ones [71], [90]. The state transition matrix in DGC indicates that the spatial dependency is stochastic depending on the restart probability and dynamic instead of being fixed. The traffic flow is related to a diffusion process on a traffic graph to model its dynamic spatial correlations. In addition, the bidirectional diffusion in DGC offers the model more flexibility to capture the influence from both upstream and downstream traffic [109]. In a word, DGC is more complicated than SGC. DGC can be adopted in both symmetric or asymmetric traffic network graph while SGC can be only utilized to process symmetric traffic graph.

Existing graph convolution theories are mainly applied on 2-D signal $X \in \mathbb{R}^{\mathbf{N} \times \mathbf{F_I}}$. However, the traffic data with both spatial and temporal attributes are usually 3-D signal $\mathcal{X} \in \mathbb{R}^{\mathbf{P} \times \mathbf{N} \times \mathbf{F_I}}$. The convolution operations need to be further generalized to 3-D signal. Equal convolution operation (e.g. SGC, DGC) with the same kernel is imposed on each time step of 3-D signal $\mathcal{X}$ in parallel [71], [93], [112], [116].

In order to enhance the performance of graph convolution in traffic tasks, many works develop various variants of SGC.

Guo *et al.* [70] redefined SGC with attention mechanism to adaptively capture the dynamic correlations in traffic network: $\Theta *_{\mathcal{G}} x \approx \sum_{k=0}^{\mathbf{K}-1} \theta_k(T_k(\tilde{\mathbf{L}}) \odot \mathbf{S})x$, where $\mathbf{S} = W_1 \odot \rho((X W_2) W_3 (W_4 X)^T + b) \in \mathbb{R}^{\mathbf{N} \times \mathbf{N}}$ is the spatial attention.

Yu *et al.* [72] generalized SGC on both spatial and temporal dimensions by scanning $\mathbf{K}$ order neighbors on graph and $\mathbf{K}_t$ neighbors on time-axis without padding. The equation is as follows:

$$\mathcal{Y}_{t,j} = \rho(\sum_{t'=0}^{\mathbf{K}_t-1} \sum_{k=0}^{\mathbf{K}-1} \sum_{i=1}^{\mathbf{F_I}} \theta_{j,t',k,i} \tilde{\mathbf{L}}^k \mathcal{X}_{t-t',i}) \quad (7)$$

where $\mathcal{X}_{t-t',i} \in \mathbb{R}^{\mathbf{N}}$ is the i^{th} feature of input $\mathcal{X}$ at time $t - t'$, $\mathcal{Y}_{t,j} \in \mathbb{R}^{\mathbf{N}}$ is the j^{th} feature of output $\mathcal{Y}$ at time t.

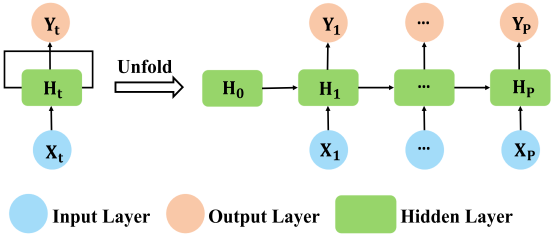


Fig. 7. The folded and unfolded structure of recurrent neural networks.

Zhao *et al.* [95] changed SGC as $\Theta *_{\mathcal{G}} x = (W \odot \tilde{\mathbf{A}}^{\mathbf{K}} \odot \mathcal{FFR})x$, where $\tilde{\mathbf{A}}^{\mathbf{K}}$ is the $\mathbf{K}$-hop neighborhood matrix and $\mathcal{FFR}$ is a matrix representing physical properties of road network. Some researchers [111], [121] followed this work and redefined $\Theta *_{\mathcal{G}} x = (W \odot Bi(\mathbf{A}^{\mathbf{K}} + \mathbf{I_N}))x$, where $Bi(.)$ is a function clipping each nonzero element in matrix to 1.

Sun *et al.* [117] modified adjacency matrix $\mathbf{A}$ in SGC as $\mathbf{S} = \mathbf{A} \odot \omega$ to integrate the geospatial positions information into the model and ω is a matrix calculated via a thresholded Gaussian kernel weighting function. The layer is built as $Y = \rho(\tilde{\mathbf{Q}}^{-\frac{1}{2}}\tilde{\mathbf{S}}\tilde{\mathbf{Q}}^{-\frac{1}{2}}XW)$, where $\tilde{\mathbf{Q}}$ is the degree matrix of $\tilde{\mathbf{S}} = \mathbf{S} + \mathbf{I_N}$.

Qiu *et al.* [58] designed a novel edge-based SGC on road network to extract the spatiotemporal correlations of the edge features. Both the feature matrix X and adjacency matrix $\mathbf{A}$ are defined on edges instead of nodes.

B. RNNs

Recurrent Neural Networks (RNNs) are a type of neural network architecture which is mainly used to detect patterns in sequential data [133]. The traffic data collected in many traffic tasks are time series data, thus RNNs are commonly utilized in these traffic tasks to capture the temporal dependency in traffic data. In this subsection, we introduce three classical models of RNNs (i.e. RNN, LSTM, GRU) and the correlations among them, providing theoretical evidence for participators to choose appropriate models for specific traffic problems.

1) RNN: Similar to a classical Feedforward Neural Network (FNN), a simple recurrent neural network (RNN) [134] contains three layers, i.e. input layer, hidden layer, output layer [135]. What differentiates RNN from FNN is the hidden layer. It passes information forward to the output layer in FNN while in RNN, it also transmits information back into itself forming a cycle [133]. For this reason, the hidden layer in RNN is called recurrent hidden layer. Such cycling trick can retain historical information, enabling RNN to process time series data.

Suppose there are $\mathbf{F_I}$, $\mathbf{F_H}$, $\mathbf{F_O}$ units in the input, hidden, output layer of RNN respectively. The input layer takes time series data $\mathbf{X} = [\mathbf{X}_1, \cdots, \mathbf{X_P}] \in \mathbb{R}^{\mathbf{P}\times\mathbf{F_I}}$ in. For each element $\mathbf{X}_t \in \mathbb{R}^{\mathbf{F_I}}$ at time t, the hidden layer transforms it to $\mathbf{H}_t \in \mathbb{R}^{\mathbf{F_H}}$ and the output layer maps $\mathbf{H}_t$ to $\mathbf{Y}_t \in \mathbb{R}^{\mathbf{F_O}}$. Note that the hidden layer not only takes $\mathbf{X}_t$ as input but also takes $\mathbf{H}_{t-1}$ as input. Such cycling mechanism enables RNN to memorize the past information (as shown in Figure 7). The mathematical notations of hidden layer and output layer are as follows:

$$\begin{aligned}
\mathbf{H}_t &= \boldsymbol{tanh}([\mathbf{H}_{t-1}, \mathbf{X}_t] \cdot W_h + b_h)\\
\mathbf{Y}_t &= \boldsymbol{\rho}(\mathbf{H}_t \cdot W_y + b_y)
\end{aligned} \tag{8}$$

where $W_h \in \mathbb{R}^{(\mathbf{F_I}+\mathbf{F_H})\times\mathbf{F_H}}$, $W_y \in \mathbb{R}^{\mathbf{F_H}\times\mathbf{F_O}}$, $b_h \in \mathbb{R}^{\mathbf{F_H}}$, $b_y \in \mathbb{R}^{\mathbf{F_O}}$ are trainable parameters. $t = 1, \cdots, \mathbf{P}$ and $\mathbf{P}$ is the input sequence length. $\mathbf{H}_0$ is initialized using small non-zero elements which can improve overall performance and stability of the network [136].

In a word, RNN takes sequential data as input and generates another sequence with the same length: $[\mathbf{X}_1, \cdots, \mathbf{X_P}] \xrightarrow{RNN} [\mathbf{Y}_1, \cdots, \mathbf{Y_P}]$. Note that we can deepen RNN through stacking multiple recurrent hidden layers.

2) LSTM: Although the hidden state enables RNN to memorize the input information over past time steps, it also introduces matrix multiplication over the (potentially very long) sequence. Small values in the matrix multiplication cause the gradients to decrease at each time step, resulting in final vanish phenomenon. Oppositely big values lead to exploding problem [137]. The vanishing or exploding gradients actually hinder the capacity of RNN to learn long-term sequential dependencies in data [135].

To overcome this hurdle, Long Short-Term Memory (LSTM) neural networks [138] are proposed to capture long-term dependency in sequence learning. Compared with the hidden layer in RNN, LSTM hidden layer has extra four parts, i.e. a memory cell, input gate, forget gate, and output gate. These three gates ranging in [0,1] can control information flow into the memory cell and preserve the extracted features from previous time steps. These simple changes enable the memory cell to store and read as much long-term information as possible. The mathematical notations of LSTM hidden layer are as follows:

$$\begin{aligned}
i_t &= \boldsymbol{\sigma}([\mathbf{H}_{t-1}, \mathbf{X}_t] \cdot W_i + b_i)\\
o_t &= \boldsymbol{\sigma}([\mathbf{H}_{t-1}, \mathbf{X}_t] \cdot W_o + b_o)\\
f_t &= \boldsymbol{\sigma}([\mathbf{H}_{t-1}, \mathbf{X}_t] \cdot W_f + b_f)\\
\mathbf{C}_t &= f_t \odot \mathbf{C}_{t-1} + i_t \odot \boldsymbol{tanh}([\mathbf{H}_{t-1}, \mathbf{X}_t] \cdot W_c + b_c)\\
\mathbf{H}_t &= o_t \odot \boldsymbol{tanh}(\mathbf{C}_t)
\end{aligned} \tag{9}$$

where i_t, o_t, f_t are the input gate, output gate, forget gate at time t respectively. $\mathbf{C}_t$ is the memory cell at time t.

3) GRU: While LSTM is a viable option for avoiding vanishing or exploding gradients, its complex structure leads to more memory requirement and longer training time. Chung *et al.* [139] proposed a simple yet powerful variant of LSTM, i.e. Gated Recurrent Unit (GRU). The LSTM cell has three gates, but the GRU cell only has two gates, resulting in fewer parameters thus shorter training time. However, GRU is equally effective as LSTM empirically [139] and is widely used in various tasks. The mathematical notations of GRU hidden layer are as follows:

$$\begin{aligned}
r_t &= \boldsymbol{\sigma}([\mathbf{H}_{t-1}, \mathbf{X}_t] \cdot W_r + b_r)\\
u_t &= \boldsymbol{\sigma}([\mathbf{H}_{t-1}, \mathbf{X}_t] \cdot W_u + b_u)\\
\tilde{\mathbf{H}}_t &= \boldsymbol{tanh}(r_t \odot [\mathbf{H}_{t-1}, \mathbf{X}_t] \cdot W_h + b_h)\\
\mathbf{H}_t &= u_t \odot \mathbf{H}_{t-1} + (1 - u_t) \odot \tilde{\mathbf{H}}_t
\end{aligned} \tag{10}$$

where r_t is the reset gate, u_t is the update gate.

4) RNNs in Traffic Domain: RNNs have shown impressive capability of processing time series data. Since traffic data has a distinct temporal dependency, RNNs are usually leveraged to capture temporal correlation in traffic data. Among the works we survey, only Geng *et al.* [90] utilized RNN to capture temporal dependency in traffic data while more than a half adopted GRU and some employed LSTM. This can be explained that RNN survives severe gradient disappearance or gradient explosion while LSTM and GRU handle this successfully and GRU can reduce the training time.

In addition, there are many tricks to augment RNNs' capacity to model the complex temporal dynamics in traffic domain, such as attention mechanism, gating mechanism and residual mechanism.

For instance, Geng *et al.* [90] incorporated the contextual information, i.e. output of SGCN which contains information of related regions, into an attention operation to model the correlations between observations at different timestamps:

$$\begin{aligned} z &= F_{pool}(\mathbf{X}_t, SGCN(\mathbf{X}_t)) \\ S &= \boldsymbol{\sigma}(W_1 \boldsymbol{ReLU}(W_2 z)) \\ \mathbf{H}_t &= RNN([\mathbf{H}_{t-1}, \mathbf{X}_t] \odot S) \end{aligned} \tag{11}$$

where $F_{pool}(\cdot)$ is a global average pooling layer, $RNN(\cdot)$ denotes the RNN hidden layer.

Chen *et al.* [113] took external factors into consideration by embedding external attributes into the input. In addition, they added the previous hidden states to the next hidden states through a residual shortcut path, which they believed can make GRU more sensitive and robust to sudden changes in traffic historical observations. The new hidden state is formulated as: $\mathbf{H}_t = GRU([\mathbf{H}_{t-1}, \mathbf{X}_t], \mathbf{E}_t) + \mathbf{H}_{t-1}W$, where $\mathbf{E}_t$ is the external features at time t, W is a linear trainable parameter, $\mathbf{H}_{t-1}W$ is the residual shortcut.

Yu *et al.* [106] inserted a dilated skip connection into GRU by changing hidden state from $\mathbf{H}_t = GRU([\mathbf{H}_{t-1}, \mathbf{X}_t])$ to $\mathbf{H}_t = GRU(\mathbf{H}_{t-s}, \mathbf{X}_t)$, where s refers to the skip length or dilation rate of each layer, $GRU(\cdot)$ denotes the GRU hidden layer. Such hierarchical design of dilation brings in multiple temporal scales for recurrent units at different layers which achieves multi-timescale modeling.

Despite the tricks above, some works replace the matrix multiplication in RNNs' hidden layer with spectral graph convolution (SGC) or diffusion graph convolution (DGC), to capture spatial-temporal correlations jointly. Take GRU as example:

$$\begin{aligned} r_t &= \boldsymbol{\sigma}([\mathbf{H}_{t-1}, \mathbf{X}_t] \ast_{\mathcal{G}} W_r + b_r) \\ u_t &= \boldsymbol{\sigma}([\mathbf{H}_{t-1}, \mathbf{X}_t] \ast_{\mathcal{G}} W_u + b_u) \\ \tilde{\mathbf{H}}_t &= \boldsymbol{tanh}(r_t \odot [\mathbf{H}_{t-1}, \mathbf{X}_t] \ast_{\mathcal{G}} W_h + b_h) \\ \mathbf{H}_t &= u_t \odot \mathbf{H}_{t-1} + (1 - u_t) \odot \tilde{\mathbf{H}}_t \end{aligned} \tag{12}$$

The $\ast_{\mathcal{G}}$ can represent SGC, DGC or other convolution operations. In the literatures we survey, most replacements happen in GRU and only one in LSTM [67]. Among GRU related traffic works, [97], [107], [109], [113], [119] replaced matrix multiplication with DGC, [43], [78], [106] with SGC, [105], [120] with GAT.

Note that besides RNNs, other techniques (e.g. TCN in the next subsection) are also popular choices to extract the temporal dynamics in traffic tasks.

C. TCN

Although RNN-based models become widespread in time-series analysis, RNNs for traffic prediction still suffer from time-consuming iteration, complex gate mechanism, and slow response to dynamic changes [93]. On the contrary, 1D-CNN has the superiority of fast training, simple structure, and no constraints to previous steps [140]. However, 1D-CNN is less common than RNNs in practice due to its lack of memory for a long sequence [141]. In 2016, a novel convolution operation integrating causal convolution and dilated convolution [142] is proposed, which outperforms RNNs in text-to-speech tasks. The prediction of causal convolution depends on previous elements but not on future elements. Dilated convolution expands the receptive field of original filter by dilating it with zeros [143]. Bai *et al.* [144] simplified the causal dilated convolution [142] for sequence modeling problem and renamed it as temporal convolution network (TCN). Recently, more and more works employ TCN to process traffic data [71], [93], [103], [112].

1) Sequence Modeling and 1-D TCN: Given an input sequence with length $\mathbf{P}$ denoted as $\mathbf{x} = [\mathbf{x}_1, \cdots, \mathbf{x}_\mathbf{P}] \in \mathbb{R}^{\mathbf{P}}$, sequence modeling aims to generate an output sequence with the same length, denoted as $\mathbf{y} = [\mathbf{y}_1, \cdots, \mathbf{y}_\mathbf{P}] \in \mathbb{R}^{\mathbf{P}}$. The key assumption is that the output at current time $\mathbf{y}_t$ only depends on historical data $[\mathbf{x}_1, \cdots, \mathbf{x}_t]$ but does not depend on any future inputs $[\mathbf{x}_{t+1}, \cdots, \mathbf{x}_\mathbf{P}]$, i.e. $\mathbf{y}_t = f(\mathbf{x}_1, \cdots, \mathbf{x}_t)$, f is the mapping function.

Obviously, RNN, LSTM and GRU can be solutions to sequence modeling tasks. However, TCN can tackle sequence modeling problem more efficiently than RNNs for that it can capture long sequence properly in a non-recursive manner. The dilated causal convolution in TCN is formulated as follows:

$$\mathbf{y}_t = \Theta \ast_{\mathcal{T}^{\mathbf{d}}} \mathbf{x}_t = \sum_{k=0}^{\mathbf{K}-1} w_k \mathbf{x}_{t-\mathbf{d}k} \tag{13}$$

where $\ast_{\mathcal{T}^{\mathbf{d}}}$ is the dilated causal operator with dilation rate $\mathbf{d}$ controlling the skipping distance, $\Theta = [w_0, \cdots, w_{\mathbf{K}-1}] \in \mathbb{R}^{\mathbf{K}}$ is the kernel. Zero padding strategy is utilized to keep the output length the same as the input length (as shown in Figure 8). Without padding, the output length is shortened by $(\mathbf{K} - 1)\mathbf{d}$ [93].

To enlarge the receptive field, TCN stacks multiple dilated causal convolution layers with $\mathbf{d} = 2^l$ as the dilation rate of l^{th} layer (as shown in Figure 8). Therefore, the receptive field in the network grows exponentially without requiring many convolutional layers or larger filter, which can handle longer sequence with less layers and save computation resources [103].

2) TCN in Traffic Domain: There are many traffic works related with sequence modeling, especially traffic spatial-temporal forecasting tasks. Compared with RNNs, the non-recursive calculation manner enables TCN to alleviate the gradient explosion problem and facilitate the training by

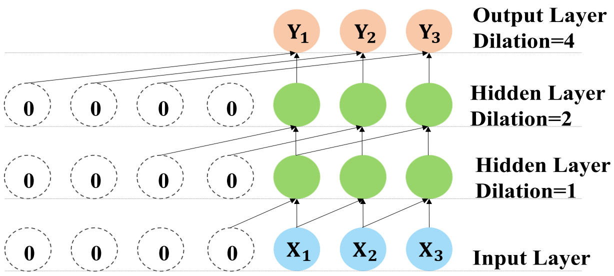


Fig. 8. Multiple dilated causal convolution layers in TCN: $[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3]$ is the input sequence and $[\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3]$ is the output sequence with the same length. The size of kernel is 2 and the dilation rate sequence is [1, 2, 4]. Zero padding strategy is taken.

parallel computation. Therefore, some works adopt TCN to capture the temporal dependency in traffic data.

Most graph-based traffic data is 3-D signal denoted as $\mathcal{X} \in \mathbb{R}^{\mathbf{P}\times\mathbf{N}\times\mathbf{F_I}}$, which requires the generalization of 1-D TCN to 3-D TCN. The dilated causal convolution can be adopted to produce the j^{th} output feature of node i at time t as follows [71]:

$$\mathcal{Y}^i_{t,j} = \rho(\Theta_j *_{\mathcal{T}^{\mathbf{d}}} \mathcal{X}^i_t) = \rho(\sum_{m=1}^{\mathbf{F_I}} \sum_{k=0}^{\mathbf{K}-1} w_{j,m,k}\mathcal{X}^i_{t-\mathbf{d}k,m}) \quad (14)$$

where $1 \leq j \leq \mathbf{F_O}$, $\mathcal{Y}^i_{t,j} \in \mathbb{R}$ is the j^{th} output feature of node i at time t. $\mathcal{X}^i_{t-\mathbf{d}k,m} \in \mathbb{R}$ is the m^{th} input feature of node i at time $t - \mathbf{d}k$. The kernel $\Theta_j \in \mathbb{R}^{\mathbf{K}\times\mathbf{F_I}}$ is trainable. $\mathbf{F_O}$ is the number of output features.

The same convolution kernel is applied to all nodes in the traffic network and each node produces $\mathbf{F_O}$ new features. The mathematical formulation of each layer is as follows [71], [112]:

$$\mathcal{Y} = \rho(\Theta *_{\mathcal{T}^{\mathbf{d}}} \mathcal{X}) \quad (15)$$

where $\mathcal{X} \in \mathbb{R}^{\mathbf{P}\times\mathbf{N}\times\mathbf{F_I}}$ represents the historical observations of the whole traffic network over past $\mathbf{P}$ time slices, $\Theta \in \mathbb{R}^{\mathbf{K}\times\mathbf{F_I}\times\mathbf{F_O}}$ represents the related convolution kernel, $\mathcal{Y} \in \mathbb{R}^{\mathbf{P}\times\mathbf{N}\times\mathbf{F_O}}$ is the output of TCN layer.

There are some tricks to enhance the performance of TCN in specific traffic tasks. For instance, Fang *et al.* [112] stacked multiple TCN layers to extract the short-term neighboring dependency by bottom layer and long-term temporal dependency by higher layer:

$$\mathcal{Y}^{(l+1)} = \sigma(\Theta^l *_{\mathcal{T}^{\mathbf{d}}} \mathcal{Y}^{(l)}) \quad (16)$$

where $\mathcal{Y}^{(l)}$ is the input of l^{th} layer, $\mathcal{Y}^{(l+1)}$ is the output and $\mathcal{Y}^{(0)} = \mathcal{X}$. $\mathbf{d} = 2^l$ is the dilation rate of l^{th} layer.

To reduce the complexity of model training, Ge *et al.* [71] constructed a residual block containing two TCN layers with the same dilation rate. The block input was added to last TCN layer to get the block output:

$$\mathcal{Y}^{(l+1)} = \mathcal{Y}^{(l)} + \boldsymbol{ReLU}(\Theta^l_1 *_{\mathcal{T}^{\mathbf{d}}} (\boldsymbol{ReLU}(\Theta^l_0 *_{\mathcal{T}^{\mathbf{d}}} \mathcal{Y}^{(l)}))) \quad (17)$$

where Θ^l_1, Θ^l_2 are the convolution kernels of the first layer and the second layer respectively. $\mathcal{Y}^{(l)}$ is the input of residual block and $\mathcal{Y}^{(l+1)}$ is its output.

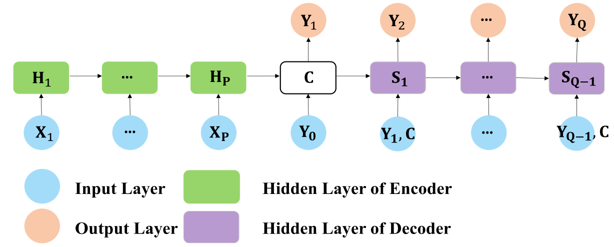


Fig. 9. Sequence to Sequence Structure without attention mechanism.

Wu *et al.* [103] integrated gating mechanism [141] with TCN to learn complex temporal dependency in traffic data:

$$\mathcal{Y} = \rho_1(\Theta_1 *_{\mathcal{T}^{\mathbf{d}}} \mathcal{X} + b_1) \odot \rho_2(\Theta_2 *_{\mathcal{T}^{\mathbf{d}}} \mathcal{X} + b_2) \quad (18)$$

where $\rho_2(\cdot) \in [0, 1]$ determines the ratio of information passed to the next layer.

Similarly, Yu *et al.* [93] used the Gated TCN and set the dilation rate $\mathbf{d} = 1$ without zero padding to shorten the output length as $\mathcal{Y} = (\Theta_1 *_{\mathcal{T}^1} \mathcal{X}) \odot \sigma(\Theta_2 *_{\mathcal{T}^1} \mathcal{X})$. They argued that this can discover variances in time series traffic data.

D. Seq2Seq

1) Seq2Seq: Sequence to Sequence (Seq2Seq) model proposed in 2014 [145] has been widely used in sequence prediction such as machine translation [146]. Seq2Seq architecture consists of two components, i.e. an encoder in charge of converting the input sequence $\mathbf{X}$ into a fixed latent vector $\mathbf{C}$, and a decoder responsible for converting $\mathbf{C}$ into an output sequence $\mathbf{Y}$ (as shown in Figure 9). Note that $\mathbf{X}$ and $\mathbf{Y}$ can have different lengths.

$$\mathbf{X}=[\mathbf{X}_1, \cdots, \mathbf{X}_i, \cdots, \mathbf{X_P}] \xrightarrow{Seq2Seq} \mathbf{Y}=[\mathbf{Y}_1, \cdots, \mathbf{Y}_j, \cdots, \mathbf{Y_Q}] \quad (19)$$

where $\mathbf{P}$ is the input length and $\mathbf{Q}$ is the output length. $\mathbf{X}_i$ is the input at time step i. $\mathbf{Y}_j$ is the output at time step j.

The specific calculation of $\mathbf{Y}_j$ is denoted as follows:

$$\begin{aligned} \mathbf{H}_i &= Encoder(\mathbf{X}_i, \mathbf{H}_{i-1}) \\ \mathbf{C} &= \mathbf{H_P}, \mathbf{S}_0 = \mathbf{H_P} \\ \mathbf{S}_j &= Decoder(\mathbf{C}, \mathbf{Y}_{j-1}, \mathbf{S}_{j-1}) \\ \mathbf{Y}_j &= \mathbf{S}_j W \end{aligned} \quad (20)$$

here, $\mathbf{H}_i$ is the hidden state of encoder. $\mathbf{H}_0$ is initialized using small non-zero elements. $\mathbf{S}_j$ is the decoder hidden state. $\mathbf{Y}_0$ is the representation of beginning sign. Note that the encoder and decoder can be any model as long as it can accept sequence and produce sequence, such as RNN, LSTM, GRU or other novel models.

A major limitation of Seq2Seq is that the latent vector $\mathbf{C}$ is fixed for each $\mathbf{Y}_j$ while $\mathbf{Y}_j$ might have stronger correlation with $\mathbf{X}_j$ than other elements. To address this issue, attention mechanism is integrated into Seq2Seq, allowing the decoder to focus on task-relevant parts of the input sequence, helping the decoder make better prediction.

$$\begin{aligned} \mathbf{H}_i &= Encoder(\mathbf{X}_i, \mathbf{H}_{i-1}) \\ \mathbf{C}_j &= \sum_{i=1}^{\mathbf{P}} (\theta_{ji} \mathbf{H}_i), \mathbf{S}_0 = \mathbf{H}_{\mathbf{P}} \\ \mathbf{S}_j &= Decoder(\mathbf{C}_j, \mathbf{Y}_{j-1}, \mathbf{S}_{j-1}) \\ \mathbf{Y}_j &= \mathbf{S}_j W \end{aligned} \tag{21}$$

where $\theta_{ji} = \frac{\exp(f_{ji})}{\sum_{k=1}^{\mathbf{P}} \exp(f_{jk})}$ is the normalized attention score, and $f_{ji} = f(\mathbf{H}_j, \mathbf{S}_{i-1})$ [146] is a function to measure the correlation between i^{th} input and j^{th} output, for instance, Luong *et al.* [147] proposed three kinds of attention score calculation.

$$f_{ji} = \begin{cases} \mathbf{H}_j^T \mathbf{S}_{i-1} & \text{dot} \\ \mathbf{H}_j^T \boldsymbol{W_a} \mathbf{S}_{i-1} & \text{general} \\ \boldsymbol{v}_a^T \tanh\left(\boldsymbol{W_a}\left[\mathbf{H}_j, \mathbf{S}_{i-1}\right]\right) & \text{concat} \end{cases} \tag{22}$$

Another way to enhance Seq2Seq performance is the scheduled sampling technique [148]. The inputs of decoder during training and testing phases are different. Decoder during training phase is fed with true labels of training datasets while it is fed with predictions generated by itself during testing phase, which accumulates error at testing time and causes degraded performance. To mitigate this issue, scheduled sampling is integrated into the model. At j^{th} iteration during the training process, the probability of feeding the decoder with true label is set as ϵ_j and the probability of feeding the decoder with prediction at the previous step is set as $1 - \epsilon_j$. Probability ϵ_j gradually decreases to 0, allowing the decoder to learn the testing distribution [109], keeping the training and testing as same as possible.

2) Seq2Seq in Traffic Domain: Since Seq2Seq can take in an input sequence and generate an output sequence with different length, it is applied on multi-step prediction in many traffic tasks. The encoder encodes the historical traffic data into a latent space vector. Then, the latent vector is fed into a decoder to generate the future traffic conditions.

Attention mechanism is usually incorporated into Seq2Seq to model the different influence on future prediction from previous traffic observations at different time slots [99], [101], [111], [77].

The encoder and decoder in many traffic literatures are in charge of capturing spatial-temporal dependencies. For instance, Li *et al.* [109] proposed DCGRU to be the encoder and decoder, which can capture spatial and temporal dynamics jointly. The design of encoder and decoder is usually the core contribution and novel part of relative works. Note that the encoder and decoder are not necessarily the same and we have made a summarization of Seq2Seq structure in previous graph-based traffic works (as shown in Table III).

RNNs-based decoder has a severe error accumulation problem during testing inference due to that each previous predicted step is the input to produce the next step prediction. The scheduled sampling to alleviate this problem is adopted in [105], [109]. RNNs-based decoder is replaced with a short-term and long-term decoder to take in last step prediction exclusively, thus easing error accumulation [77]. The utilization of Seq2Seq technique in traffic domain is flexible. For instance, Seq2Seq is integrated into a bigger framework, being the generator and discriminator of GAN [101].

TABLE III

THE ENCODERS AND DECODERS OF SEQUENCE TO SEQUENCE ARCHITECTURE

References	Encoder	Decoder
[109]	GRU+DGCN	Same as encoder
[101]	SGCN +LSTM	LSTM+SGCN
[99]	STAtt Block	Same as encoder
[42]	MLPs	An MLP
[110]	SGCN+Pooling+GRU	GCN+Upooling+GRU
[105]	GRU with graph self-attention	Same as encoder
[43]	GRU+SGCN	Same as encoder
[111]	SGCN+ bidirectional GRU	Same as encoder
[77]	Long-term encoder (Gated SGCN)	Short-term encoder
[114]	SGCN+LSTM	LSTM
[97]	SGCN+GRU	Same as encoder
[78]	CGRM (GRU, SGCN)	Same as encoder
[104]	LSTM+RGC	RGC
[49]	LSTM	Same as encoder

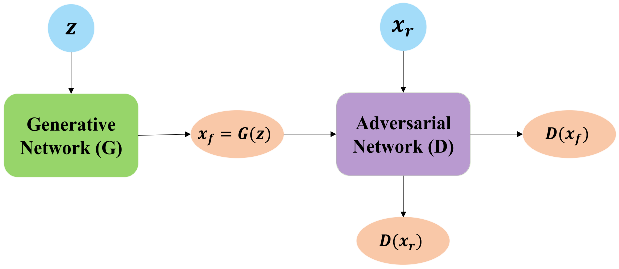


Fig. 10. Generative Adversarial Network: Generator G is in charge of producing a generated sample $x_f = G(z)$ from a random vector z, which is sampled from a prior distribution p_z. Discriminator D is in charge of discriminating between the fake sample x_f generated from G and the real sample x_r from the training data.

E. GAN

1) GAN: Generative Adversarial Network (GAN) [149] is a powerful deep generative model aiming to generate artificial samples as indistinguishable as possible from their real counterparts. GAN, inspired by game theory, is composed of two players, a generative neural network called Generator G and an adversarial network called Discriminator D (as shown in Figure 10).

Discriminator D tries to determine whether the input samples belong to the generated data or the real data while Generator G tries to cheat on Discriminator D by producing samples as true as possible. The two mutually adversarial and optimized processes are alternately trained, which strengthens the performance of both D and G. When the fake sample produced by G is very close to the ground truth and D is unable to distinguish them any more, it is considered that Generator G has learned the true distribution of the real data and the model converges. At this time, we can consider this game to reach a Nash equilibrium [150].

Mathematically, such process can be formulated to minimize their losses $Loss_G$ and $Loss_D$. With the loss function being cross entropy denoted as f, we can have:

$$\begin{aligned} Loss_G &= f(D(G(z)), 1) = -\sum \log D(G(z)) \\ \phi^* &= \underset{\phi}{\operatorname{argmin}}(Loss_G) = \underset{\phi}{\operatorname{argmax}}(-Loss_G) \\ &= \underset{\phi}{\operatorname{argmax}} \mathbb{E}(\log D(G(z))) \end{aligned} \tag{23}$$

$$\begin{aligned} Loss_D &= f(D(x_r), 1, D(x_f), 0) \\ &= -\sum \log D(x_r) - \sum \log(1 - D(x_f)) \\ \theta^* &= \underset{\theta}{\operatorname{argmin}}(Loss_D) = \underset{\theta}{\operatorname{argmax}}(-Loss_D) \\ &= \underset{\theta}{\operatorname{argmax}}(\mathbb{E}(\log D(x_r) + \log(1 - D(x_f)))) \end{aligned} \tag{24}$$

where 1 is the label of true sample x_r. 0 is the label of fake sample $x_f = G(z)$. ϕ and θ are the trainable parameters of G and D respectively. Note that when G is trained, D is untrainable. Interested readers can refer to [151], [152] for surveys of GAN.

2) GAN in Traffic Domain: When GAN is applied in traffic prediction tasks [153], [154], Generator G is usually employed to generate future traffic observations based on the historical observations. Then the generated data and the future real data are fed into Discriminator D to train it. After training, Generator G can learn the distribution of the real traffic flow data through a large number of historical data and can be used to predict the future traffic states [101]. GAN can be also utilized to solve the sparsity problem of traffic data for its efficacy in handling data generation [96].

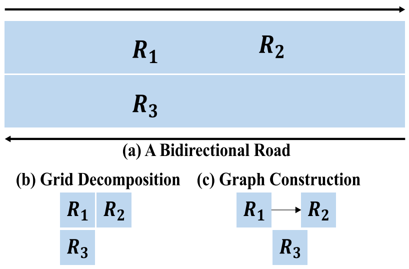
In addition, the generator or discriminator of GAN can be any model, such as RNNs, Seq2Seq, depending on the specific traffic tasks.

VI. Challenges Perspective

Traffic tasks are very challenging due to the complicated spatial dependency, temporal dependency in traffic data. In addition, external factors such as holiday or event can also affect the traffic conditions. In this section, we introduce four common challenges in traffic domain. We carefully examine each challenge and its corresponding solutions, making necessary comparison.

A. Spatial Dependency

As mentioned in previous section, some literatures [131], [132], [155] extract spatial features through decomposing the whole traffic network into grids and then employing CNNs to process the grid-based data. However, the grid-based assumption actually violates the nature topology of traffic network. Many traffic networks are physically organized as a graph and the graph topology information is obviously valuable for traffic prediction (as shown in Figure 11). According to our survey, graph neural networks can model spatial dependencies in graph-based traffic networks much better than grid-based approaches. In addition, the complicated spatial dependencies in traffic network can be categorized into three spatial attributes, i.e. spatial locality, multiple relationships and global connectivity. Different kinds of GNNs combining with other deep learning techniques are utilized to solve different kinds of spatial attributes.



(a) A Bidirectional Road

(b) Grid Decomposition (c) Graph Construction

Fig. 11. The formulation of a bidirectional road: The traffic condition of road R_1 is only influenced by the same side road R_2 and has weak correlation with the opposite side road R_3. But if this region is modeled as grids, R_3 has similar impact on R_1 as R_2, which is against the truth. If it is model as a graph, R_1 is connected with R_2 and disconnected with R_3, which can reflect the true relationship.

1) Spatial Locality: Spatial locality refers that adjacent regions are usually highly relevant to each other. For example, the passenger flow of a station in a subway is obviously affected by its connected stations. $\mathbf{K}$-localized spectral graph convolution network (SGCN) is widely adopted to aggregate the information of 0 to $\mathbf{K} - 1$ hop neighbors to the central region. In addition, some works make different assumptions about the spatial locality and utilize some novel tricks.

The adjacency matrix representing the traffic topology is usually pre-defined while some works [43], [70] argued that neighboring locations are dynamically correlated with each other. They incorporated the attention mechanism into SGCN to adaptively capture the dynamic correlations among surrounding regions.

SGCN requires all the regions to have the same local statistics and its convolution kernel is location-independent. However, Zhang *et al.* [69] clarified that the local statistics of traffic data changed from region to region and they designed location-dependent kernels for different regions automatically.

2) Multiple Relationships: While locality attribute focuses on spatial proximity, the target region can be correlated with distant regions through various non-Euclidean relationships such as functional similarity, transportation connectivity (as shown in Figure 5), semantic neighbors. Functional similarity refers that distant region is similar to the target region in terms of functionality, which can be characterized by the surrounding POIs [71], [90]. Transportation connectivity suggests that those geographically distant but conveniently reachable can be correlated [90]. The reachable way can be motorway, highway, subway. Semantic neighbors are adopted to model the correlation between origins and destinations [102]. The correlation is measured by the passenger flow between them. To explicitly extract these correlation information, different types of correlations using multiple graphs are encoded [90] and multi-graph convolution is leveraged.

3) Global Connectivity: Both spatial proximity and multi-relationship focus on parts of the network while ignore the whole structure. Global connectivity refers that traffic conditions of different regions have influenced each other at a whole network scale. There are several strategies to exploit the global structure information of traffic network.

A popular way to capture global connectivity is to model the changing traffic conditions in the traffic network as a diffusion process that happens at a network scale, which is presented by a power series of transition matrices. Then, diffusion graph convolution network (DGCN) is adopted to extract the spatial dependency globally [101], [103], [109], [113], [97], [119].

A novel spatial graph pooling layer with path growing algorithm is designed to produce a coarser graph [106]. This pooling layer is stacked before SGC layer to get multi-granularity graph convolutions, which can extract spatial features at various scopes.

A SGC layer with a self-adaptive adjacency matrix is proposed [103] to capture the hidden global spatial dependency in the data. This self-adaptive adjacency matrix is learned from the data through an end-to-end supervised training.

B. Temporal Dependency

Temporal dependency refers that prediction of traffic conditions at a certain time is usually correlated with various historical observations [93].

As stated in Section V, many works extract the temporal dependency by RNNs-based approaches. However, RNNs-based approaches suffer from time-consuming iterations and confront gradient vanishing/explosion problem for capturing long sequence. Compared with RNNs-based approaches, TCN-based approaches have the superiority of simple structures, parallel computing and stable gradients. Therefore, some works [71], [93] adopt TCN-based approaches to capture the temporal pattern in traffic data. In addition, TCN is able to handle different temporal levels by stacking multiple layers. For instance, Fang *et al.* [112] and Wu *et al.* [103] stacked multiple TCN layers with the bottom layers extracting short-term neighboring dependencies and the higher layers learning long-term temporal patterns.

1) Multi-Timescale: Some works extract the temporal dependency at a multi-timescale perspective [70], [117]. Temporal dependency is decomposed into recent, daily and weekly dependencies [70]. The recent dependency refers that the future traffic conditions are influenced by the traffic conditions recently. For instance, the traffic congestion at 9 am inevitably influences traffic flow at the following hours. Daily dependency describes that the repeated daily pattern in traffic data due to the regular daily routine of people, such as morning peak and evening peak. Weekly dependency considers the influence caused by the same week attributes. For instance, all Mondays share similar traffic pattern in a short-term. Guo *et al.* [70] set three parallel components with the same structure to model these three temporal attributes respectively.

2) Different Weights: Some works argue that the correlations between historical and future observations are varying at different previous time slices. Guo *et al.* [70] adopted a temporal attention mechanism to adaptively attach different importance to historical data.

C. Spatiotemporal Dependency

Many works capture the spatial and temporal dependency separately in a sequential manner [66], [92], [95], [101], [111], [89], [121] while the spatial and temporal dependencies are closely intertwined in traffic data. Guo *et al.* [70] argued that the historical observations in different locations at different times have varying impacts on central region in the future. Take an obvious example, a traffic accident in a critical road results in serious disruptions over related roads but at different time, due to the gradual formation and dispersion of traffic congestion.

A limitation of separately modeling is that the potential interactions between spatial features and temporal features are ignored, which may hurt the prediction performance. To overcome such limitation, a popular way is to incorporate the graph convolution operations (e.g. SGC, DGC) to RNNs (as stated in Section VI) to capture spatial-temporal correlations jointly [67], [107], [109], [113], [97], [119], [43], [78], [106].

D. External Factors

Factors such as holidays, time attributes (e.g. hour, day, week, month, season, year) [71], [117], weather (e.g. rainfall, temperature, air quality) [117], special events, POIs [90] and traffic incidents (e.g. incident time, incident type) [92] can influence the traffic prediction in some extent, which we refer as external factors or context factors. In addition, Zhang *et al.* [111] considered historical statistical speed information (e.g. average or standard deviation of traffic speed) as external factor.

Some factors such as day attributes, holidays and weather conditions are encoded as discrete values and they are usually transformed into binary vectors by one-hot encoding. Other factors including temperature, wind speed are encoded as continual values and they are usually normalized by Min-Max normalization or Z-score normalization.

There are two approaches to handle external factors in the literatures we survey. The first approach is to concatenate the external factors with other features and feed them into model [71], [113]. The second approach is to design an external component in charge of processing external factors alone. The external component usually contains two fully connected layers, of which the first extracting important features and the second mapping low dimension features to high dimension features [71], [92], [117], [49]. Bai *et al.* [114] employed multi-LSTM layers to extract representation of external factors. The output of external component is fused with other components to generate the final result.

VII. Public Datasets and Open Source Codes

A. Public Datasets

We summarize some public datasets (as shown in Table IV) in the literatures we survey to help successors participate in this domain and produce more valuable works.

B. Open Source Codes

Open-source implementations are helpful for researchers to compare their approaches. We provide the hyperlinks of public source codes of the literatures reviewed in this paper (as shown in Table V) to facilitate the baseline experiments in traffic domain.

TABLE IV
SOME OPEN TRAFFIC DATASETS

Datasets	Links	References
NYC taxi	https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page	[100], [117], [92], [104]
NYC bike	https://www.citibikenyc.com/system-data	[117], [49], [77], [114]
San Francisco taxi	https://crawdad.org/ crawdad/epfl/mobility/20090224/	[92]
Chicago bike	https://www.divvybikes.com/system-data	[49]
BikeDC (Bike Washington)	https://www.capitalbikeshare.com/system-data	[117]
California -PEMS	http://pems.dot.ca.gov/	[93], [71], [70], [100], [113], [72], [99], [103], [107], [67], [97]

TABLE V
SOME OPEN SOURCE CODES

Reference	Model	Year	Framework	Github
[109]	DCRNN	2018	Tensorflow	https://github.com/liyaguang/DCRNN
[98]	GCNN	2018	Keras	https://github.com/RingBDStack/GCNN-In-Traffic
[94]	T-GCN	2019	Tensorflow	https://github.com/lehaifeng/T-GCN
[99]	GMAN	2019	Tensorflow	https://github.com/zhengchuanpan/GMAN
[103]	Graph-WaveNet	2019	Torch	https://github.com/nnzhan/Graph-WaveNet

VIII. FUTURE DIRECTIONS

We have investigated the latest advances in graph-based traffic literatures and made a summary of these literatures in Table II. Further, we suggest some directions for researchers to explore, which can be divided into three categories, i.e. application related, technique related, external factor related directions.

1) Application Related Directions: As shown in Table II, there are many works utilizing graph-based deep learning architectures to tackle traffic state prediction and traffic demand prediction, which have achieved state-of-the-art performance. However, there are only a handful of works analyzing traffic data in a graph perspective in other research directions, such as vehicle behavior classification [66], optimal dynamic electronic toll collection (DETC) scheme [58], path availability [67], traffic signal control [68]. When it comes to traffic incident detection, vehicle detection, origin-destination travel demand prediction and transfer learning from City to City, works adopting graph-based deep learning techniques are rare up to now. Therefore, the upcoming participators can explore these directions in a graph perspective and learn the successful experiences from existing works.

2) Technique Related Directions: On one hand, most existing works have employed spectral graph convolution network (SGCN) and diffusion graph convolution network (DGCN), two popular kinds of GNNs, to analyze traffic tasks. There are only a handful of works utilizing Graph attention networks (GATs) in traffic domain [99], [105], [108], [120], [123]. Other kinds of GNNs, such as graph autoencoders (GAEs) [156], [157], recurrent graph neural networks (RecGNNs) [158] have achieved state-of-the-art performance in other domains, but they are seldom explored in traffic domain up to now. Therefore, it is worth to extend these branches of GNNs to traffic domain. On the other hand, recent works have combined GNNs with other deep learning techniques such as RNNs, TCN, Seq2Seq, GAN to solve the challenges in traffic tasks. However, few traffic works consider transfer learning, continue learning and reinforcement learning together with GNNs, which might be a promising direction for researchers. In addition, most of the graph-based traffic works are regression tasks, while classification tasks are few [66], [67]. Researchers can explore the classification traffic tasks in a graph perspective.

3) External Factors Related Directions: Finally, many existing traffic models do not take external factors into consideration, for that external factors are hard to collect and have various formats. The data sparsity of external factors is still a challenge confronted by the research community. In addition, the techniques to process external factors are rather naive, e.g. a simple fully connected layer. There should be more approaches to process external factors.

IX. CONCLUSION

In this survey, we conduct a comprehensive review of various graph-based deep learning architectures in recent traffic works. More specifically, we summarize a general graph-based formulation of traffic problem and graph construction from various traffic datasets. Further, we decompose all the investigated architectures and analyze the common modules they share, including graph neural networks (GNNs), recurrent neural networks (RNNs), temporal convolution network (TCN), Sequence to Sequence (Seq2Seq) model, generative adversarial network (GAN). We provide a thorough description of their variants in traffic tasks, hoping to provide upcoming researchers insights into how to design novel techniques for their own traffic tasks. We also summarize the common challenges in many traffic scenarios, such as spatial dependency, temporal dependency, external factors. More than that, we present multiple deep learning based solutions for each challenge. In addition, we provide some hyperlinks of public datasets and codes in related works to facilitate the upcoming researches. Finally, we suggest some future directions for participators interested in this domain.

ACKNOWLEDGMENT

The authors would like to thank anonymous reviewers for their valuable comments.

REFERENCES

[1] G. Yu and C. Zhang, “Switching ARIMA model based forecasting for traffic flow,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2004, pp. 429–432.

[2] 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, Nov. 2003.

[3] S. R. Chandra and H. Al-Deek, “Predictions of freeway traffic speeds and vol. using, vector autoregressive models,” *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 53–72, May 2009.

[4] Y. Xie, Y. Zhang, and Z. Ye, “Short-term traffic volume forecasting using Kalman filter with discrete wavelet decomposition,” *Comput.-Aided Civil Infrastruct. Eng.*, vol. 22, no. 5, pp. 326–334, Jul. 2007.

[5] H. Fu, H. Ma, Y. Liu, and D. Lu, “A vehicle classification system based on hierarchical multi-SVMs in crowded traffic scenes,” *Neurocomputing*, vol. 211, pp. 182–190, Oct. 2016.

[6] M. May, D. Hecker, C. Körner, S. Scheider, and D. Schulz, “A vector-geometry based spatial kNN-algorithm for traffic frequency predictions,” in *Proc. IEEE Int. Conf. Data Mining Workshops*, Dec. 2008, pp. 442–447.

[7] J. Liu, T. Li, P. Xie, S. Du, F. Teng, and X. Yang, “Urban big data fusion based on deep learning: An overview,” *Inf. Fusion*, vol. 53, pp. 123–133, Jan. 2020.

[8] Z. Lv, J. Xu, K. Zheng, H. Yin, P. Zhao, and X. Zhou, “LC-RNN: A deep learning model for traffic speed prediction,” in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3470–3476.

[9] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, “Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction,” *Sensors*, vol. 17, no. 4, p. 818, Apr. 2017.

[10] L. Yan, H. Shen, J. Zhao, C. Xu, F. Luo, and C. Qiu, “CatCharger: Deploying wireless charging lanes in a metropolitan road network through categorization and clustering of vehicle traffic,” in *Proc. IEEE INFOCOM - IEEE Conf. Comput. Commun.*, May 2017, pp. 1–9.

[11] Y. Sun, X. Yu, R. Bie, and H. Song, “Discovering time-dependent shortest path on traffic graph for drivers towards green driving,” *J. Netw. Comput. Appl.*, vol. 83, pp. 204–212, Apr. 2017.

[12] H. Sun, J. Wu, D. Ma, and J. Long, “Spatial distribution complexities of traffic congestion and bottlenecks in different network topologies,” *Appl. Math. Model.*, vol. 38, no. 2, pp. 496–505, Jan. 2014.

[13] M. Gori, G. Monfardini, and F. Scarselli, “A new model for learning in graph domains,” in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2, Jul. 2005, pp. 729–734.

[14] F. Scarselli, M. Gori, A. Chung Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[15] M. Henaff, J. Bruna, and Y. LeCun, “Deep convolutional networks on graph-structured data,” 2015, *arXiv:1506.05163*. [Online]. Available: http://arxiv.org/abs/1506.05163

[16] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, “Learning deep generative models of graphs,” 2018, *arXiv:1803.03324*. [Online]. Available: http://arxiv.org/abs/1803.03324

[17] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, “Multi-label image recognition with graph convolutional networks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5177–5186.

[18] Z. Guo, Y. Zhang, and W. Lu, “Attention guided graph convolutional networks for relation extraction,” in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 241–251.

[19] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, and E. Bombarell, “Convolutional networks on graphs for learning molecular fingerprints,” in *Proc. NIPS*, 2015, pp. 2224–2232.

[20] J. Ye, J. Zhao, K. Ye, and C. Xu, “Multi-graph convolutional network for relationship-driven stock movement prediction,” 2020, *arXiv:2005.04955*. [Online]. Available: http://arxiv.org/abs/2005.04955

[21] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, “Graph convolutional neural networks for Web-scale recommender systems,” in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 974–983.

[22] M. G. Karlaftis and E. I. Vlahogianni, “Statistical methods versus neural networks in transportation research: Differences, similarities and some insights,” *Transp. Res. C, Emerg. Technol.*, vol. 19, no. 3, pp. 387–399, Jun. 2011.

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

[24] P. Xie, T. Li, J. Liu, S. Du, X. Yang, and J. Zhang, “Urban flow prediction from spatiotemporal data using machine learning: A survey,” *Inf. Fusion*, vol. 59, pp. 1–12, Jul. 2020.

[25] H. Nguyen, L.-M. Kieu, T. Wen, and C. Cai, “Deep learning methods in transportation domain: A review,” *IET Intell. Transp. Syst.*, vol. 12, no. 9, pp. 998–1004, Nov. 2018.

[26] Y. Wang, D. Zhang, Y. Liu, B. Dai, and L. H. Lee, “Enhancing transportation systems via deep learning: A survey,” *Transp. Res. C, Emerg. Technol.*, vol. 99, pp. 144–163, Feb. 2019.

[27] M. Veres and M. Moussa, “Deep learning for intelligent transportation systems: A survey of emerging trends,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 8, pp. 3152–3168, Aug. 2020.

[28] Q. Chen *et al.*, “A survey on an emerging area: Deep learning for smart city data,” *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 5, pp. 392–410, Oct. 2019.

[29] S. Wang, J. Cao, and P. Yu, “Deep learning for spatio-temporal data mining: A survey,” *IEEE Trans. Knowl. Data Eng.*, early access, Sep. 22, 2020, doi: 10.1109/TKDE.2020.3025580.

[30] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, “Geometric deep learning: Going beyond Euclidean data,” *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[31] J. Zhou *et al.*, “Graph neural networks: A review of methods and applications,” 2018, *arXiv:1812.08434*. [Online]. Available: http://arxiv.org/abs/1812.08434

[32] J. Zhang, “Graph neural networks for small graph and giant network representation learning: An overview,” 2019, *arXiv:1908.00187*. [Online]. Available: http://arxiv.org/abs/1908.00187

[33] P. Quan, Y. Shi, M. Lei, J. Leng, T. Zhang, and L. Niu, “A brief review of receptive fields in graph convolutional networks,” in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. WI Companion*, Oct. 2019, pp. 106–110.

[34] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, “Graph convolutional networks: A comprehensive review,” *Comput. Social Netw.*, vol. 6, no. 1, p. 11, Dec. 2019.

[35] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 24, 2020, doi: 10.1109/TNNLS.2020.2978386.

[36] Y.-y. Chen, Y. Lv, Z. Li, and F.-Y. Wang, “Long short-term memory model for traffic congestion prediction with online open data,” in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 132–137.

[37] L. Yan and H. Shen, “TOP: Optimizing vehicle driving speed with vehicle trajectories for travel time minimization and road congestion avoidance,” *ACM Trans. Cyber Phys. Syst.*, vol. 4, no. 2, pp. 17:1–17:25, 2020.

[38] L. Yan, H. Shen, and K. Chen, “MobiT: Distributed and congestion-resilient trajectory-based routing for vehicular delay tolerant networks,” *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1078–1091, Jun. 2018.

[39] X. Ma, H. Yu, Y. Wang, and Y. Wang, “Large-scale transportation network congestion evolution prediction using deep learning theory,” *PLoS ONE*, vol. 10, no. 3, Mar. 2015, Art. no. e0119044.

[40] F. Sun, A. Dubey, and J. White, “DxNAT—Deep neural networks for explaining non-recurring traffic congestion,” in *Proc. Big Data*, 2017, pp. 2141–2150.

[41] L. Yan, H. Shen, and K. Chen, “MobiT: A distributed and congestion-resilient trajectory based routing algorithm for vehicular delay tolerant networks,” in *Proc. 2nd Int. Conf. Internet-Things Design Implement.*, Pittsburgh, PA, USA, Apr. 2017, pp. 209–214.

[42] L. Wei *et al.*, “Dual graph for traffic forecasting,” *IEEE Access*, early access, Dec. 9, 2019, doi: 10.1109/ACCESS.2019.2958380.

[43] W. Chen, L. Chen, Y. Xie, W. Cao, Y. Gao, and X. Feng, “Multi-range attentive bicomponent graph convolutional network for traffic forecasting,” in *Proc. AAAI*, 2020, pp. 3529–3536.

[44] Z. Cao, S. Jiang, J. Zhang, and H. Guo, “A unified framework for vehicle rerouting and traffic light control to reduce traffic congestion,” *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 7, pp. 1958–1973, Jul. 2017.

[45] L. Qi, M. Zhou, and W. Luan, “A two-level traffic light control strategy for preventing incident-based urban traffic congestion,” *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 13–24, Jan. 2018.

[46] J. Ye, J. Zhao, K. Ye, and C. Xu, “Multi-STGCnet: A graph convolution based spatial-temporal framework for subway passenger flow forecasting,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[47] F. Rodrigues, I. Markou, and F. C. Pereira, “Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach,” *Inf. Fusion*, vol. 49, pp. 120–129, Sep. 2019.

[48] D. Wang, W. Cao, J. Li, and J. Ye, “DeepSD: Supply-demand prediction for online car-hailing services using deep neural networks,” in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 243–254.

[49] D. Chai, L. Wang, and Q. Yang, “Bike flow prediction with multi-graph convolutional networks,” in *Proc. 26th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Nov. 2018, pp. 397–400.

[50] L. Lin, Z. He, and S. Peeta, “Predicting station-level hourly demand in a large-scale bike-sharing network: A graph convolutional neural network approach,” *Transp. Res. C, Emerg. Technol.*, vol. 97, pp. 258–276, Dec. 2018.

[51] X. Han, T. Grubenmann, R. Cheng, S. C. Wong, X. Li, and W. Sun, “Traffic incident detection: A trajectory-based approach,” in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Apr. 2020, pp. 1866–1869.

[52] Z. Zhang, Q. He, J. Gao, and M. Ni, “A deep learning approach for detecting traffic accidents from social media data,” *Transp. Res. C, Emerg. Technol.*, vol. 86, pp. 580–596, Jan. 2018.

[53] M. Sameen and B. Pradhan, “Severity prediction of traffic accidents with recurrent neural networks,” *Appl. Sci.*, vol. 7, no. 6, p. 476, Jun. 2017.

[54] S. Alkheder, M. Taamneh, and S. Taamneh, “Severity prediction of traffic accident using an artificial neural network,” *J. Forecasting*, vol. 36, no. 1, pp. 100–108, Jan. 2017.

[55] A. Kashevnik, I. Lashkov, and A. Gurtov, “Methodology and mobile application for driver behavior analysis and accident prevention,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2427–2436, Jun. 2020.

[56] M. Hänninen, “Bayesian networks for maritime traffic accident prevention: Benefits and challenges,” *Accident Anal. Prevention*, vol. 73, pp. 305–312, Dec. 2014.

[57] B.-W. Jo, Y.-S. Lee, R. Khan, J.-H. Kim, and D.-K. Kim, “Robust construction safety system (RCSS) for collision accidents prevention on construction sites,” *Sensors*, vol. 19, no. 4, p. 932, Feb. 2019.

[58] W. Qiu, H. Chen, and B. An, “Dynamic electronic toll collection via multi-agent deep reinforcement learning with edge-based graph convolutional networks,” in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4568–4574.

[59] H. Li, P. Wang, and C. Shen, “Toward end-to-end car license plate detection and recognition with deep neural networks,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 3, pp. 1126–1136, Mar. 2019.

[60] X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan, “Vehicle detection in satellite images by hybrid deep convolutional neural networks,” *IEEE Geosci. Remote Sens. Lett.*, vol. 11, no. 10, pp. 1797–1801, Oct. 2014.

[61] S. Zhang, J. Yang, and B. Schiele, “Occluded pedestrian detection through guided attention in CNNs,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6995–7003.

[62] H. Tayara, K. G. Soo, and K. T. Chong, “Vehicle detection and counting in high-resolution aerial images using convolutional regression neural network,” *IEEE Access*, vol. 6, pp. 2220–2230, 2018.

[63] J. Li, X. Liang, S. Shen, T. Xu, J. Feng, and S. Yan, “Scale-aware fast R-CNN for pedestrian detection,” *IEEE Trans. Multimedia*, vol. 20, no. 4, pp. 985–996, Apr. 2018.

[64] M. A. S. Kamal, T. Hayakawa, and J.-I. Imura, “Development and evaluation of an adaptive traffic signal control scheme under a mixed-automated traffic scenario,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 2, pp. 590–602, Feb. 2020.

[65] Y. Sun, T. He, J. Hu, H. Huang, and B. Chen, “Socially-aware graph convolutional network for human trajectory prediction,” in *Proc. IEEE 3rd Inf. Technol., Netw., Electron. Autom. Control Conf. (ITNEC)*, Mar. 2019, pp. 325–333.

[66] S. Mylavarapu, M. Sandhu, P. Vijayan, K. M. Krishna, B. Ravindran, and A. Namboodiri, “Towards accurate vehicle behaviour classification with multi-relational graph convolutional networks,” 2020, *arXiv:2002.00786*. [Online]. Available: http://arxiv.org/abs/2002.00786

[67] J. Li *et al.*, “Predicting path failure in time-evolving graphs,” in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1279–1289.

[68] T. Nishi, K. Otaki, K. Hayakawa, and T. Yoshimura, “Traffic signal control based on reinforcement learning with graph convolutional neural nets,” in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 877–883.

[69] Q. Zhang, Q. Jin, J. Chang, S. Xiang, and C. Pan, “Kernel-weighted graph convolutional network: A deep learning approach for traffic forecasting,” in *Proc. 24th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2018, pp. 1018–1023.

[70] 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*, 2019, pp. 922–929.

[71] L. Ge, H. Li, J. Liu, and A. Zhou, “Temporal graph convolutional networks for traffic speed prediction considering external factors,” in *Proc. 20th IEEE Int. Conf. Mobile Data Manage. (MDM)*, Jun. 2019, pp. 234–242.

[72] B. Yu, M. Li, J. Zhang, and Z. Zhu, “3D graph convolutional networks with temporal graphs: A spatial information free framework for traffic forecasting,” 2019, *arXiv:1903.00919*. [Online]. Available: http://arxiv.org/abs/1903.00919

[73] J. Hu, C. Guo, B. Yang, and C. S. Jensen, “Stochastic weight completion for road networks using graph convolutional networks,” in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Apr. 2019, pp. 1274–1285.

[74] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, “When will you arrive? Estimating travel time based on deep neural networks,” in *Proc. AAAI*, 2018, pp. 1–8.

[75] L. Yan *et al.*, “Employing opportunistic charging for electric taxicabs to reduce idle time,” *Proc. ACM Interact. Mobile Wearable Ubiquitous Technol.*, vol. 2, no. 1, pp. 47:1–47:25, 2018.

[76] X. Geng, X. Wu, L. Zhang, Q. Yang, Y. Liu, and J. Ye, “Multi-modal graph interaction for multi-graph convolution network in urban spatiotemporal forecasting,” 2019, *arXiv:1905.11395*. [Online]. Available: http://arxiv.org/abs/1905.11395

[77] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Q. Z. Sheng, “STG2Seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting,” in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 1981–1987.

[78] L. Liu, J. Chen, H. Wu, J. Zhen, G. Li, and L. Lin, “Physical-virtual collaboration modeling for intra-and inter-station metro ridership prediction,” 2020, *arXiv:2001.04889*. [Online]. Available: http://arxiv.org/abs/2001.04889

[79] Z. Li, N. D. Sergin, H. Yan, C. Zhang, and F. Tsung, “Tensor completion for weakly-dependent data on graph for metro passenger flow prediction,” in *Proc. AAAI*, 2020, pp. 4804–4810.

[80] J. Ye, J. Zhao, L. Zhang, C. Xu, J. Zhang, and K. Ye, “A data-driven method for dynamic OD passenger flow matrix estimation in urban metro systems,” in *Proc. 9th Int. Conf., Which Held Services Conf. Fed.-BigData*, Honolulu, HI, USA, vol. 12402. Cham, Switzerland: Springer, Sep. 2020, pp. 116–126.

[81] H. Shi *et al.*, “Predicting origin-destination flow via multi-perspective graph convolutional network,” in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Apr. 2020, pp. 1818–1821.

[82] J. Hu, B. Yang, C. Guo, C. S. Jensen, and H. Xiong, “Stochastic origin-destination matrix forecasting using dual-stage graph convolutional, recurrent neural networks,” in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Apr. 2020, pp. 1417–1428.

[83] K.-F. Chu, A. Y. S. Lam, and V. O. K. Li, “Deep multi-scale convolutional LSTM network for travel demand and origin-destination predictions,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 8, pp. 3219–3232, Aug. 2020.

[84] J. Li, H. Ma, Z. Zhang, and M. Tomizuka, “Social-WaGDAT: Interaction-aware trajectory prediction via wasserstein graph double-attention network,” 2020, *arXiv:2002.06241*. [Online]. Available: http://arxiv.org/abs/2002.06241

[85] A. Monti, A. Bertugli, S. Calderara, and R. Cucchiara, “DAG-net: Double attentive graph neural network for trajectory forecasting,” 2020, *arXiv:2005.12661*. [Online]. Available: http://arxiv.org/abs/2005.12661

[86] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, “Social-STGCNN: A social spatio-temporal graph convolutional neural network for human trajectory prediction,” in *Proc. CVPR*, 2020, pp. 14412–14420.

[87] J. Li, F. Yang, M. Tomizuka, and C. Choi, “EvolveGraph: Multi-agent trajectory prediction with dynamic relational reasoning,” 2020, *arXiv:2003.13924*. [Online]. Available: http://arxiv.org/abs/2003.13924

[88] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. D. Reid, H. Rezatofighi, and S. Savarese, “Social-BiGAT: Multimodal trajectory forecasting using bicycle-GAN and graph attention networks,” in *Proc. NIPS*, 2019, pp. 137–146.

[89] Z. Zhao, H. Fang, Z. Jin, and Q. Qiu, “GISNet:Graph-based information sharing network for vehicle trajectory prediction,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–7.

[90] G. Xu, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, and Y. Liu, “Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 3656–3663.

[91] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, “Time-series extreme event forecasting with neural networks at uber,” in *Proc. ICML*, vol. 34, 2017, pp. 1–5.

[92] Q. Xie, T. Guo, Y. Chen, Y. Xiao, X. Wang, and B. Y. Zhao, "How do urban incidents affect traffic speed?' A deep graph convolutional network for incident-driven traffic speed prediction," 2019, *arXiv:1912.01242*. [Online]. Available: http://arxiv.org/abs/1912.01242

[93] 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.*, Jul. 2018, pp. 3634–3640.

[94] 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. 2019.

[95] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4883–4894, Nov. 2020.

[96] J. J. Q. Yu and J. Gu, "Real-time traffic speed estimation with graph convolutional generative autoencoder," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3940–3951, Oct. 2019.

[97] Y. Huang, Y. Weng, S. Yu, and X. Chen, "Diffusion convolutional recurrent neural network with rank influence learning for traffic forecasting," in *Proc. TrustCom*, 2019, pp. 678–685.

[98] J. Li *et al.*, "Graph CNNs for urban traffic passenger flows prediction," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Oct. 2018, pp. 29–36.

[99] C. Zheng, X. Fan, C. Wang, and J. Qi, "GMAN: A graph multi-attention network for traffic prediction," in *Proc. AAAI*, 2020, pp. 1234–1241.

[100] 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*, 2019, pp. 890–897.

[101] Y. Zhang, S. Wang, B. Chen, and J. Cao, "GCGAN: Generative adversarial nets with graph CNN for network-scale traffic prediction," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.

[102] Y. Wang, H. Yin, H. Chen, T. Wo, J. Xu, and K. Zheng, "Origin-destination matrix prediction via graph convolution: A new perspective of passenger demand modeling," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1227–1235.

[103] 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.*, Aug. 2019, pp. 1907–1913.

[104] J. Ke, X. Qin, H. Yang, Z. Zheng, Z. Zhu, and J. Ye, "Predicting origin-destination ride-sourcing demand with a spatio-temporal encoder-decoder residual multi-graph convolutional network," 2019, *arXiv:1910.09103*. [Online]. Available: http://arxiv.org/abs/1910.09103

[105] Z. Kang, H. Xu, J. Hu, and X. Pei, "Learning dynamic graph embedding for traffic flow forecasting: A graph self-attentive method," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Oct. 2019, pp. 2570–2576.

[106] B. Yu, H. Yin, and Z. Zhu, "ST-UNet: A spatio-temporal U-Network for graph-structured time series modeling," 2019, *arXiv:1903.05631*. [Online]. Available: http://arxiv.org/abs/1903.05631

[107] K. Guo *et al.*, "Optimized graph convolution recurrent neural network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, early access, Jan. 14, 2020, doi: 10.1109/TITS.2019.2963722.

[108] C. Zhang, J. J. Q. Yu, and Y. Liu, "Spatial-temporal graph attention networks: A deep learning approach for traffic forecasting," *IEEE Access*, vol. 7, pp. 166246–166256, 2019.

[109] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. ICLR*, 2018.

[110] M. Lu, K. Zhang, H. Liu, and N. Xiong, "Graph hierarchical convolutional recurrent neural network (GHCRNN) for vehicle condition prediction," 2019, *arXiv:1903.06261*. [Online]. Available: http://arxiv.org/abs/1903.06261

[111] Z. Zhang, M. Li, X. Lin, Y. Wang, and F. He, "Multistep speed prediction on traffic networks: A deep learning approach considering spatio-temporal dependencies," *Transp. Res. C, Emerg. Technol.*, vol. 105, pp. 297–322, Aug. 2019.

[112] S. Fang, Q. Zhang, G. Meng, S. Xiang, and C. Pan, "GSTNet: Global spatial-temporal network for traffic flow prediction," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 2286–2293.

[113] C. Chen *et al.*, "Gated residual recurrent graph neural networks for traffic prediction," in *Proc. AAAI*, 2019, pp. 485–492.

[114] L. Bai, L. Yao, S. S. Kanhere, X. Wang, W. Liu, and Z. Yang, "Spatio-temporal graph convolutional and recurrent networks for citywide passenger demand prediction," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 2293–2296.

[115] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. AAAI*, 2018, pp. 7444–7452.

[116] K. Chen *et al.*, "Dynamic spatio-temporal graph-based CNNs for traffic flow prediction," *IEEE Access*, vol. 8, pp. 185136–185145, 2020.

[117] J. Sun, J. Zhang, Q. Li, X. Yi, Y. Liang, and Y. Zheng, "Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks," *IEEE Trans. Knowl. Data Eng.*, early access, Jul. 13, 2020, doi: 10.1109/TKDE.2020.3008774.

[118] Y. Zhang, T. Cheng, and Y. Ren, "A graph deep learning method for short-term traffic forecasting on large road networks," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 34, no. 10, pp. 877–896, Oct. 2019.

[119] X. Zhou, Y. Shen, and L. Huang, "Revisiting flow information for traffic prediction," 2019, *arXiv:1906.00560*. [Online]. Available: http://arxiv.org/abs/1906.00560

[120] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D. Yeung, "GaAN: Gated attention networks for learning on large and spatiotemporal graphs," in *Proc. UAI*, 2018, pp. 339–349.

[121] Y. Shin and Y. Yoon, "Incorporating dynamicity of transportation network with multi-weight traffic graph convolutional network for traffic forecasting," *IEEE Trans. Intell. Transp. Syst.*, early access, Oct. 26, 2020, doi: 10.1109/TITS.2020.3031331.

[122] A. Majumdar, "Graph structured autoencoder," *Neural Netw.*, vol. 106, pp. 271–280, Oct. 2018.

[123] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. ICLR*, 2018.

[124] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NIPS*, 2017, pp. 1024–1034.

[125] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[126] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. ICLR*, 2014.

[127] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. NIPS*, 2016, pp. 3837–3845.

[128] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2011.

[129] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2017.

[130] S.-H. Teng, "Scalable algorithms for data and network analysis," *Found. Trends Theor. Comput. Sci.*, vol. 12, nos. 1–2, pp. 1–274, 2016.

[131] J. Zhang, Y. Zheng, and D. Qi, "Deep spatio-temporal residual networks for citywide crowd flows prediction," in *Proc. AAAI*, 2017, pp. 1655–1661.

[132] S. Du, T. Li, X. Gong, and S. Horng, "A hybrid method for traffic flow forecasting using multimodal deep learning," *Int. J. Comput. Intell. Syst.*, vol. 13, no. 1, pp. 85–97, 2020.

[133] R. M. Schmidt, "Recurrent neural networks (RNNs): A gentle introduction and overview," 2019, *arXiv:1912.05911*. [Online]. Available: http://arxiv.org/abs/1912.05911

[134] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[135] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," 2017, *arXiv:1801.01078*. [Online]. Available: http://arxiv.org/abs/1801.01078

[136] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. ICML*, vol. 28. Apr. 2013, pp. 1139–1147.

[137] G. Chen, "A gentle tutorial of recurrent neural network with error backpropagation," 2016, *arXiv:1610.02583*. [Online]. Available: http://arxiv.org/abs/1610.02583

[138] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[139] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. NIPS Workshop*, 2014.

[140] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," 2016, *arXiv:1610.10099*. [Online]. Available: http://arxiv.org/abs/1610.10099

[141] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. ICML*, vol. 70, 2017, pp. 933–941.

[142] A. van den Oord *et al.*, "WaveNet: A generative model for raw audio," in *Proc. ISCA Workshop*, 2016, pp. 124–125.

[143] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proc. ICLR*, 2016.

[144] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*. [Online]. Available: http://arxiv.org/abs/1803.01271

[145] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014, pp. 3104–3112.

[146] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR*, 2015.

[147] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1412–1421.

[148] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proc. NIPS*, 2015, pp. 1171–1179.

[149] I. Goodfellow, J. Pouget-Abadie, M. Mirza, and E. Xu, "Generative adversarial nets," in *Proc. NIPS*, 2014, pp. 2672–2680.

[150] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Proc. NIPS*, 2017, pp. 6626–6637.

[151] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Process.*, vol. 35, no. 1, pp. 53–65, Jan. 2017.

[152] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, "Generative adversarial networks: Introduction and outlook," *IEEE/CAA J. Automatica Sinica*, vol. 4, no. 4, pp. 588–598, Sep. 2017.

[153] Y. Lin, X. Dai, L. Li, and F.-Y. Wang, "Pattern sensitive prediction of traffic flow based on generative adversarial framework," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 6, pp. 2395–2400, Jun. 2019.

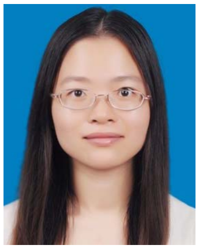
[154] Y. Liang, Z. Cui, Y. Tian, H. Chen, and Y. Wang, "A deep generative adversarial architecture for network-wide spatial-temporal traffic-state estimation," *Transp. Res. Rec. J. Transp. Res. Board*, vol. 2672, no. 45, pp. 87–105, Dec. 2018.

[155] H. Yao *et al.*, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Proc. AAAI*, 2018, pp. 2588–2595.

[156] A. Hasanzadeh, E. Hajiramezanali, K. R. Narayanan, N. Duffield, M. Zhou, and X. Qian, "Semi-implicit graph variational auto-encoders," in *Proc. NIPS*, 2019, pp. 10711–10722.

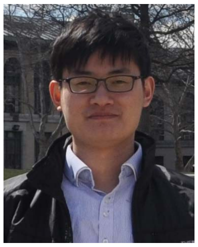
[157] M. Simonovsky and N. Komodakis, "GraphVAE: Towards generation of small graphs using variational autoencoders," in *Proc. Int. Conf. Artif. Neural Netw.*, 2018, pp. 412–422.

[158] H. Dai, Z. Kozareva, B. Dai, A. Smola, and L. Song, "Learning steady-states of iterative algorithms over graphs," in *Proc. ICML*, 2018, pp. 1106–1114.



Jiexia Ye received the bachelor's degree of economics in finance from Sun Yat-sen University in 2012. She is currently pursuing the master's degree of engineering in computer science with the University of Chinese Academy of Sciences. Her research interests include graph neural networks/graph representation learning in traffic and finance domain.

Juanjuan Zhao (Member, IEEE) received the Ph.D. degree from the Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences, in 2017, and the M.S. degree from the Department of Computer Science, Wuhan University of Technology, in 2009. She is currently an Assistant Professor with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. Her research interests include data-driven urban systems, mobile data collection, cross-domain data fusion, and heterogeneous model integration.



Kejiang Ye (Member, IEEE) received the B.Sc. and Ph.D. degrees in computer science from Zhejiang University in 2008 and 2013, respectively. He was a Joint Ph.D. Student with The University of Sydney from 2012 to 2013. After graduation, he worked as a Post-Doctoral Researcher with Carnegie Mellon University from 2014 to 2015 and Wayne State University from 2015 to 2016. He is currently an Associate Professor with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Science. His research interests include cloud computing, big data, and network systems.

Chengzhong Xu (Fellow, IEEE) received the Ph.D. degree from The University of Hong Kong, China, in 1993. He was a Chief Scientist of the Shenzhen Institutes of Advanced Technology (SIAT), Chinese Academy of Sciences, and the Director of the Institute of Advanced Computing and Digital Engineering, SIAT. He was also with Wayne State University, USA, for 18 years. He is currently the Dean of the State Key Laboratory of IOTSC, Department of Computer Science, University of Macau, Macau, and a Chair Professor of computer science with UM. He has published over 300 peer-reviewed articles on these topics with over 10K citations. His research interests include parallel and distributed systems, cloud and edge computing, and data-driven intelligence. He is also the Chair of IEEE Technical Committee of Distributed Processing. He has served in the editorial boards for leading journals, including IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and *Journal of Parallel and Distributed Computing*. He is also the Associate Editor-in-Chief of *ZTE Communication*.