Highlights

- A novel deep learning model for network-wide traffic prediction
- Designed framework for extracting hidden and dynamic spatial-temporal features
- A virtual road graph used for further spatial feature extraction and correction
- Use of two real-world traffic datasets in the evaluation of the proposed model

VDGCNeT: A Novel Network-wide Virtual Dynamic Graph Convolution Neural Network and Transformer-based Traffic Prediction Model

Ge Zheng[a], Wei Koong Chai[b], Jiankang Zhang[b], Vasilis Katos[b]

[a]*Institute of Manufacturing (IfM), Department of Engineering, University of Cambridge, 17 Charles Babbage Road, Cambridge, CB3 0FS, Cambridgeshire, United Kingdom*

[b]*Department of Computing and Informatics, Bournemouth University, Fern Barrow, Poole, BH12 5BB, Dorset, United Kingdom*

Abstract

We address the problem of traffic prediction on large-scale road networks. We propose a novel deep learning model, Virtual Dynamic Graph Convolution Neural Network and Transformer with Gate and Attention mechanisms (VDGCNeT), to comprehensively extract complex, dynamic and hidden spatial dependencies of road networks for achieving high prediction accuracy. For this purpose, we advocate the use of a virtual dynamic road graph that captures the dynamic and hidden spatial dependencies of road segments in real road networks instead of purely relying on the physical road connectivity. We further design a novel framework based on Graph Convolution Neural Network (GCN) and Transformer to analyze dynamic and hidden spatial-temporal features. The gate mechanism is utilized for concatenating learned spatial and temporal features from Spatial and Temporal Transformers, respectively, while the Attention-based Similarity is used to update dynamic road graph. Two real-world traffic datasets from large-scale road networks with different properties are used for training and testing our model. We compare our VDGCNeT against nine other well-known models in the literature. Our results demonstrate that the proposed VDGCNeT is capable of achieving highly accurate predictions – on average 96.77% and 91.68% accuracy on `PEMS-BAY` and `METR-LA` datasets respectively. Overall, our VDGCNeT performs the best when compared against other existing models.

Keywords: Traffic Prediction, Deep Learning, Virtual Road Graph, Intelligent Transportation System, Large-Scale Road Networks

1. Introduction

It is projected that by mid-21st century, the world's urban population will almost double from over 3.4 billion in 2009 to 6.4 billion in 2050. The worst hit areas by traffic congestion are typically urban areas. Worsening traffic situation needs to be improved urgently. Intelligent Transportation Systems (ITSs) [1] aim to offer innovative transport and traffic management services. With emergence of new sensor and communication technologies, more data are collected and shared between various ITS applications. This enables ITS to provide timely traffic management functions to improve the efficiency of road networks, reduce traffic congestion and improve road safety.

In this paper, we focus on traffic prediction which is an important function of ITSs. It aims to predict future traffic states based on historical traffic states. It helps drivers to avoid traffic congestion by informing them the traffic states in the road ahead in advance and also provides useful insights to traffic management departments for devising efficient traffic management strategies. Since the earliest work that used Auto-Regressive Moving Average (ARMA) to predict traffic volume and occupancy [2], various works on traffic prediction have emerged. Broadly, we see three distinct phases in the evolution of existing works: 1) the first phase mostly employs statistical methods [2, 3, 4, 5]; 2) the second phase uses machine learning approaches [6, 7, 8]; 3) the third phase advocates advanced deep learning models [9, 10, 11, 12]. The main breakthrough that led the transition from the first to second phase is that machine learning models with non-linear kernels / activation functions can efficiently analyze non-linear relations of traffic in time domain while statistical models largely fail to capture such non-linear relations of traffic data [13]. With new sensor technologies, traffic data can be collected with more features. Deep learning models, capable of learning features from large high-dimensional datasets, are then proposed, leading the methods of solving traffic prediction problems to the third phase; overcoming the disadvantage of machine learning models that are shallow and often, insufficient for analyzing high-dimensional traffic data.

Early deep learning models, e.g., Deep Belief Network (DBN) [9][10], Recurrent Neural Network (RNN) [11] and its variants Long Short-Term Memory (LSTM) [12] and Gated Recurrent Unit (GRU) [14] and Convolutional Neural Network (CNN) [15], were proposed to extract temporal re-

lationship of traffic data. However, [16] indicated that, besides temporal relationship, road networks are also spatially correlated. Thus, sharing information among neighboring sensors can improve prediction accuracy. To concurrently consider both temporal and spatial relationships of traffic data on an entire road network, these deep learning models are then combined to form ensemble models. For example, both [17] and [18] developed CNN-based ensemble models to learn traffic patterns by first converting traffic data to images before computing their predictions. FCL-Net [19], another ensemble model, stacks and fuses multiple convolutional long short-term memory layers, standard LSTM layers and convolutional layers. A Spatial-Temporal Attentive Neural Network (STANN) [20] was developed based on the encoder-decoder architecture with attention mechanisms for traffic speed prediction, in which LSTMs with attention mechanisms are used in both encoder and decoder. Meanwhile, Lui [21] introduced a LSTM module with time-aware attention for modeling long-term features and a convolution module for modeling spatial-temporal features towards time-aware location prediction. Further, [22] designed an ensemble model exploiting LSTM, Deep AutoEncoder (DAE) and CNN to improve prediction accuracy while [23] developed a spatial-temporal dynamic network using local CNN and LSTM to respectively handle spatial and temporal information. These ensemble models tend to perform better than individual deep learning models.

The aforementioned models consider road networks as regular grids and traffic data having regular Euclidean structure. However, road networks are inherently irregular and traffic data may be non-Euclidean [24]. To overcome these, the Graph Convolutional Network (GCN) [25] was introduced. GCN allows convolutional operations on non-Euclidean data to obtain the relationships of traffic data in the space domain. Following this, [26] developed the Diffusion Convolutional Recurrent Neural Network (DCRNN) for traffic prediction on a road network graph, while [27] built the STGCN model that consists of two spatial-temporal convolutional blocks and a fully-connected output layer for network-wide traffic prediction. The TGC-LSTM model [28] uses GCN to capture the spatial dependencies of neighboring road segments and LSTM to extract temporal dependencies. Their ensuing work [29] proposed to use graph wavelet, instead of GCN, to extract spatial features. STAWnet [30] is proposed for multi-step traffic prediction using self-attention network for the dynamic spatial dependency analysis and temporal convolution for the long temporal dependency analysis. An end-to-end global spatial-temporal graph attention network (GST-GAT) proposed by [31] uses

"global interaction + node query" to model the dynamic spatial-temporal relations of traffic data. Furthermore, [32] embedded attention mechanism into GCN to obtain dynamic spatial features by assigning a probability to a road segment so as to contribute to the targeted road segment for traffic prediction, while [33] designed a novel structure based on GCN, named GC-TrellisNetsED, to capture the spatial correlation among metro stations and also the dynamics of such correlations, working with a temporal module for multi-step metro station passenger flow prediction.

The most recent works, that commonly use GCN to extract spatial dependencies of traffic on road networks, usually consider actual physical road connections between road segments. This requires topological data of the road network (conventionally represented by an adjacency matrix). Since the adjacency matrix only contains information on connections between adjacent neighbors, $k-hop$ matrix built based on the adjacency matrix is sometimes used to extract connectivity information within a fixed local neighborhood. These do not comprehensively encode the complex spatial dependencies hidden within the road network. The traffic congestion spreads not only to its adjacent road segments but also to a local area, and this local area is not fixed and changes over time [34]. Therefore, a fixed adjacency matrix or $k-hop$ matrix is insufficient to fully describe the complex and dynamic spatial dependencies in the road network.

To account for these, *virtual dynamic graph* is introduced in this work to explore hidden spatial dependencies. There are several existing works using dynamic GCNs for traffic prediction. For example, [35] built a dynamic GCN framework for passenger flow prediction using historical passenger flows to model the relationships of traffic stations. Later focusing on long-term traffic flow prediction, [36] adopted reinforcement learning approach and proposed the graph convolutional policy network for generating dynamic graphs when the dynamic graphs are incomplete due to data sparsity issue. In this paper, we advocate the concept of virtual dynamic graphs and propose a novel deep learning model named, Virtual Dynamic Graph Convolution Network and Transformer with Gate and Attention mechanisms (VDGCNeT) for traffic prediction on large-scale road networks. The main contributions of this paper are as follows:

- Our VDGCNeT model makes predictions using a virtual dynamic road network updated after each batch based on the similarity of historical and targeted traffic data. This is inspired by the application of GCN

on solving classification problems [37]. Instead of purely relying on the physical road topology, we consider a number of randomly selected traffic data for each node once. We then treat the historical traffic data in those samples as the node features to analyze the relations of each node to others on the network for generating and updating the virtual dynamic graph. Not only can this method capture dynamic and hidden correlations among road segments across the network, it also allows the model to learn a graph to describe the area affected by each road segment differently in the used road network (cf. Section 3.1). From time domain, the learned graph can describe summarized patterns over the whole past several months by continuously updating the learned graph over the fed data in the training process. From space domain, traffic state of each road segment in the learned graph depends on all other road segments differently. This means that all different influences by other road segments to the targeted road segment are considered. Therefore, we consider that the trained model is more general and can cover all different situations.

- We exploit the attention mechanism of the transformer technology [38] to capture dynamic and hidden spatial and temporal features from historical traffic data. We employ graph convolution neural network to correct the learned dynamic spatially-fused features from transformers on the updated dynamic graph (cf. Section 3.2).

- We compare our VDGCNeT model against nine well-known existing models using two real-world large-scale traffic datasets. We further conduct ablation experiments to gain insights into the characteristics of our model by systematically evaluating it with removal of individual constituent module.

The rest of this paper is organized as follows. Firstly, we formulate the traffic prediction problem in Section 2. Then we present in detail our VDGCNeT model and the design rationale in Section 3. We discuss our evaluation results in Section 4. Finally, we conclude our work in Section 5.

2. Problem Formulation

Considering a road network with a set of N geographically distributed sensors, x_t^i denotes the traffic speed measured at node i at t^{th} time interval.

The traffic speed data is written as $x_t = \{x_t^1, x_t^2, \ldots, x_t^i, \ldots, x_t^{N-1}, x_t^N\}; x_t \in \mathbb{R}^N$. Typical time interval, m, could be 5, 15, 30, 45 and 60 mins [39]. The datasets used for our experiments are with $m = 5$ mins. We then define X_T and X_D as the traffic speed data collected from N sensors for T previous time intervals and the same time interval with the targeted time interval in D previous days, respectively. They are formulated as follows:

$$\begin{aligned} X_T = \{x_1, x_2, \ldots, x_t, \ldots, x_T\}; \\ X_T \in \mathbb{R}^{T \times N}, T = 1, 2, 3, \ldots \end{aligned} \tag{1}$$

$$\begin{aligned} X_D = \{X_{(T+T')-\frac{24\times60}{m}\times D}, X_{(T+T')-\frac{24\times60}{m}\times(D-1)}, \ldots, \\ X_{(T+T')-\frac{24\times60}{m}\times 2}, X_{(T+T')-\frac{24\times60}{m}\times 1}\}; \\ X_D \in \mathbb{R}^{D\times T'\times N}, D = 1, 2, 3, \ldots, T' = 1, 2, 3 \ldots \end{aligned} \tag{2}$$

In a similar manner, future traffic data is denoted as $X_{T+T'} = \{x_{T+1}, x_{T+2}, \ldots, x_{T+t'}, \ldots, x_{T+T'}\} \in \mathbb{R}^{T'\times N}$ where T' is the prediction horizon. In this paper, we consider multi-interval predictions where $T' = \{1, 2, 3, \ldots, 12\}$ corresponding to $\{5, 10, 15, \ldots, 60\}$ minutes. These are common values used in the literature (e.g., [26][28]).

Conventionally (e.g., [40][41]), the road graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes representing road segments or sensor locations with $|\mathcal{V}| = N$ and $\mathcal{E}$ is the set of edges representing connectivity between road segments. $\mathcal{G}$ can be represented by $A \in \mathbb{R}^{N\times N}$, the $N \times N$ symmetric adjacency matrix, with its element $A_{i,j} = 1$ if there exists a link between node i and j, otherwise $A_{i,j} = 0$. Since future traffic state of a node is influenced by its own current state, $\mathcal{G}$ can then be written as $A_0 = (A + I_N) \in \mathbb{R}^{N\times N}$ where I_N is the $N \times N$ identity matrix. However, in our work, instead of purely using physical road connectivity, we advocate the use of virtual dynamic road graph (cf. Section 3). The virtual dynamic road graph generated by the Att-Similar Block in our model is represented by $A_{vd} \in \mathbb{R}^{N\times N}$. Considering that traffic speed has both short- and long-term temporal patterns [42], the timestamps of traffic data X_T, including the time interval of a day and the day of a week, are defined as external features (i.e., $TT' = \{t_T, t_D\}; TT' \in \mathbb{R}^{(T+T')\times 2}$, $t_T = \{t_1, \ldots, t_t, \ldots, t_T, t_{T+1}, \ldots, t_{T+t'}, \ldots, t_{T+T'}\} \in \mathbb{R}^{T+T'}; t_t = 1, 2, \ldots, \frac{24\times60}{m}$ and $t_D = \{t_{d_1}, \ldots, t_{d_t}, \ldots, t_{d_T}, t_{d_{T+1}}, \ldots, t_{d_{T+t'}}, \ldots, t_{d_{T+T'}}\} \in \mathbb{R}^{T+T'}; t_{dt} = 1, 2, 3, 4, 5, 6, 7$, respectively) and are used to embed external

temporal features. The timestamps of traffic data X_D are defined as $TDT' \in \mathbb{R}^{D\times(T+T')\times 2}$ in a similar manner. Based on traffic speed data and the road graph information, the traffic prediction problem considered here can be formulated as following.

$$\tilde{X}_{T+T'} = F\Big(X_T; X_D; TT'; TDT'; \mathcal{G}(\mathcal{V}, \mathcal{E}, A_0)\Big) \tag{3}$$

where the objective is to learn the mapping function $F(.)$ and then use the learned $F(.)$ to compute the traffic speed in the next T' time intervals based on traffic speed data in T previous time intervals and in the same time interval with the targeted time interval from D previous days, their timestamps and the virtual road graph information.

3. Virtual Dynamic Graph Convolution Network and Transformer with Gate and Attention mechanisms (VDGCNeT)

3.1. VDGCNeT Workflow

Fig. 1 presents the workflow of VDGCNeT for network-wide traffic prediction given a road map. It predicts traffic in the next T' time intervals using historical traffic data. It consists of two main phases (i.e., training and testing phases) with each phase taking slightly different inputs.

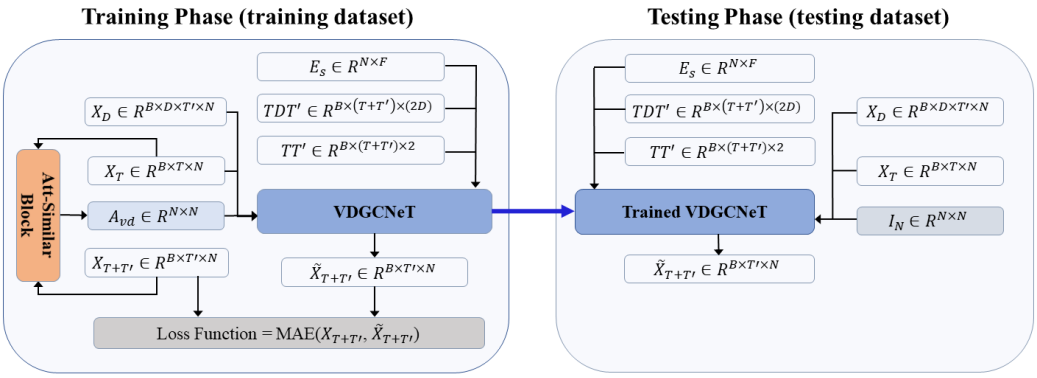


Figure 1: The workflow of the VDGCNeT model for multi-interval traffic predictions.

In the training phase, VDGCNeT takes five inputs:

1. historical traffic data for T time intervals, $X_T \in \mathbb{R}^{B\times T\times N}$ (Note that X_T is formatted to $\mathbb{R}^{B\times T\times N}$ for the training process and the same operation applies to other inputs. B is the batch size.),

2. historical traffic data in the same time interval with the targeted time interval from past D days, $X_D \in \mathbb{R}^{B\times D\times T'\times N}$,

3. the timestamps of time intervals in a day and days in a week, $TT' \in \mathbb{R}^{B\times(T+T')\times 2}$ and $TDT' \in \mathbb{R}^{B\times(T+T')\times 2D}$,

4. pre-embedded physical spatial matrix based on the original road graph, $\mathcal{G}(\mathcal{V}, \mathcal{E}, A_0)$,

5. virtual dynamic spatial matrix, $A_{vd} \in \mathbb{R}^{N\times N}$.

Particularly, the virtual dynamic spatial matrix, A_{vd}, is updated after each batch in the training phase by a similarity block that is built based on the idea of the attention mechanism, named Att-Similar-Block. This enables VDGCNeT to learn the dynamic and hidden spatial dependencies of road segments across the network. This Att-Similar-Block maps and represents the relationships between historical traffic data X_T and $X_{T+T'}$ via the following:

$$A_{vd} = Softmax(Reshape(X_T)(Reshape(X_{T+T'}))^{\intercal}) \tag{4}$$

where $Reshape(.)$ is used to reshape $\mathbb{R}^{B\times T(orT')\times N}$ into $\mathbb{R}^{N\times BT(orBT')}$ so as to obtain the virtual dynamic spatial matrix $A_{vd} \in \mathbb{R}^{N\times N}$. This operation enables each node to build the relationships with others through BT features from B samples. It results in the learned spatial weight matrix that is (1) dynamic as it is updated after each batch in the process and (2) general as the learned spatial weight matrix is not specific to any time interval due to fact that we use randomly selected B samples rather than particular fixed time intervals.

After the training phase, the spatial dependencies of road segments on the network would have been learned and encoded in the weight and bias matrices of A_{vd}. When the trained VDGCNeT model is tested, the virtual dynamic spatial matrix, A_{vd}, is replaced by the identity matrix, $I_N \in \mathbb{R}^{N\times N}$. This aims to restore the obtained hidden spatial dependencies, that were learned from historical data, by multiplying the learned spatial weight matrix by an identity matrix and then use it for testing. This is because, mathematically, any parameter matrix multiplied by the Identity matrix yields the parameter matrix itself. In addition, it can also avoid information leakage and thus ensure that model predictions are made only based on learned features and historical traffic data.

3.2. VDGCNeT Model Architecture

Since VDGCNeT uses the same internal structure to extract features from X_T and X_D, we will only detail in the following the internal structure for X_T as depicted in Fig. 2. It consists of three main blocks, i.e., two Spatial and Temporal Transformers blocks (STTras-Blocks), each consisting of a Spatial Transformer (STm), a Temporal Transformer (TTm) and a Spatial-Temporal Fusion (STFm) module, and a Dynamic Graph Convolution Network Block (DGCN-Block), connected between the two STTras-Blocks. The STTras-Blocks are used for spatial and temporal feature analysis. The DGCN-Block is for dynamic spatial feature analysis as well as for correcting the learned spatially-fused feature from the first STTras-Block. In addition, VDGCNeT also includes two additional embedding modules, namely Spatial Embedding (SEm) for spatial matrix (E_s) embedding based on the physical road network, A_0, and Temporal Embedding (TEm) for temporal matrix (T_T and $T_{T'}$) embedding based on the timestamps of traffic data, TT'. The Att-Similar-Block is included only in the training phase for updating the virtual dynamic spatial matrix A_{vd} which takes place after each batch.

The embedded spatial matrix, E_s, is generated by SEm using the physical road graph, A_0, while the embedded temporal matrices, T_T and $T_{T'}$, are generated by TEm using the timestamps of the historical and targeted traffic data TT'. X_T and E_s are sent to STm in the first STTras-Block for spatial feature learning while X_T and T_T are fed to its TTm for historical temporal feature learning. Both learned spatial and temporal features are fused in STFm based on the gate mechanism in GRU [43]. Then the output with the virtual dynamic spatial matrix, A_{vd}, are sent to the DGCN-Block for further dynamic spatial feature analysis and learned spatially-fused feature correction. The output of DGCN-Block, A_{vd}, E_s and $T_{T'}$, are further fed into the second STTras-Block for learning dynamic spatial and temporal features by aligning the learned features from DGCN-Block with virtual dynamic spatial matrix A_{vd} in STm and with the targeted temporal embedding matrix $T_{T'}$ in TTm. Finally, the fused dynamic spatial-temporal features from STFm are sent to a Fully-Connected Layer for the final prediction. We detail the operations of each VDGCNeT block and module next.

3.2.1. SEm

This module embeds the physical road network into a spatial matrix E_s using the *node2vec* algorithm [44]. It maps nodes in the network to a low-dimensional space while keeping the node relationships to their neighbors.

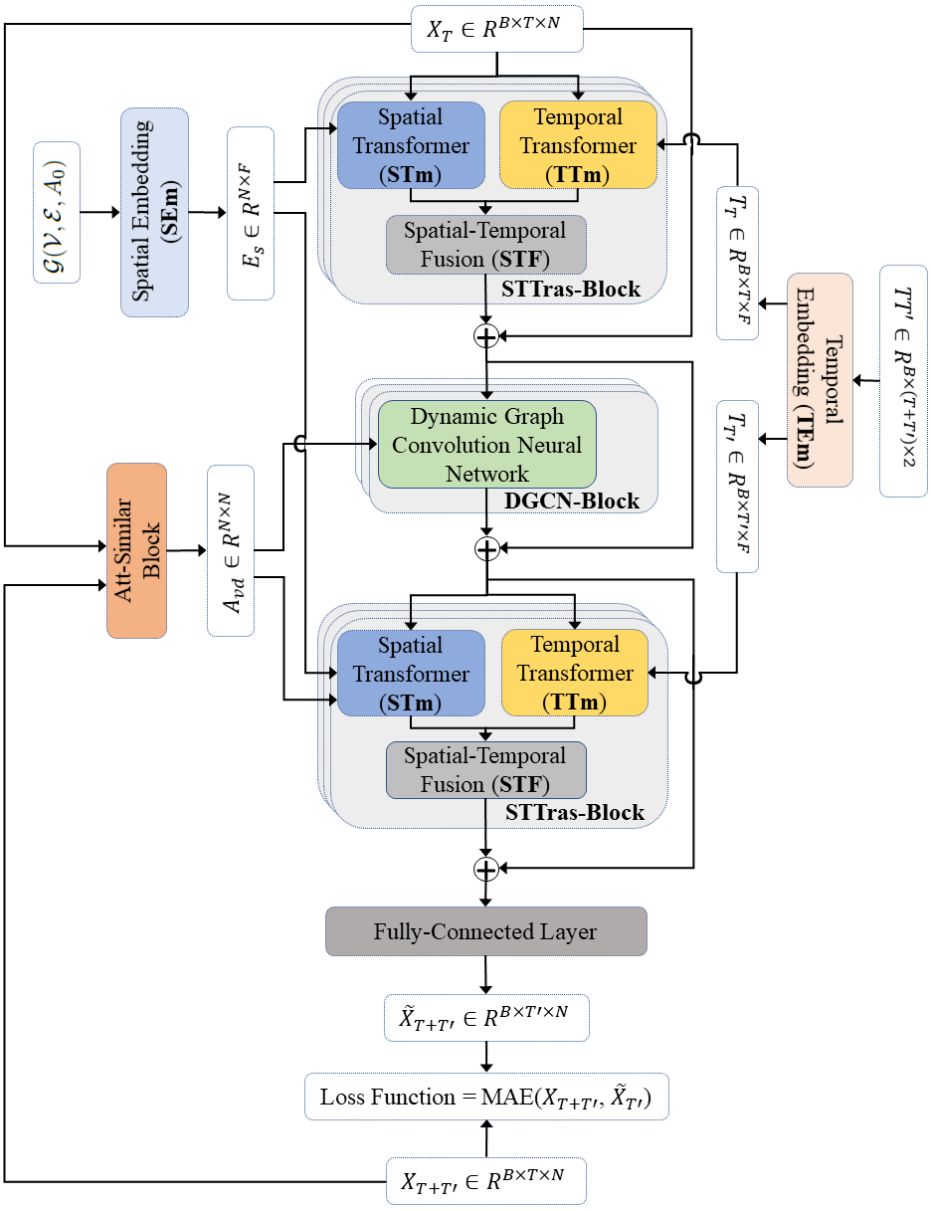


Figure 2: Internal structure of the VDGCNeT model.

In our work, the physical road network, $A_0 \in \mathbb{R}^{N\times N}$, is learned and mapped as $E_s \in \mathbb{R}^{N\times F}; F < N$ by a biased random walk procedure, which provides a flexible neighborhood size to each node for learning richer representations.

3.2.2. TEm

This module embeds the timestamps of historical and targeted traffic data as external temporal features. It consists of an encoding layer and a fully-connected layer. The encoding layer applies the One-Hot Encoding technique [45] to embed $TT' \in \mathbb{R}^{B\times(T+T')\times 2}$ as $TT'_{em} \in \mathbb{R}^{B\times(T+T')\times 295}$ where 295 is the sum of 288 (i.e., the time-interval timestamps in a day) and 7 (i.e., the number of daily timestamps in a week). Hence, one value in the third dimension used to present the time-interval timestamp from TT' is embedded into a vector of length 288 while the other one for the daily timestamp is embedded into a vector of length 7. Both embedded vectors are concatenated. Then, TT'_{em} is sent to a fully-connected layer before being split into $T_T \in \mathbb{R}^{B\times T\times F}$ as historical-external temporal features and $T_{T'} \in \mathbb{R}^{B\times T'\times F}$ as targeted-external temporal features where F is the number of embedded features in the Fully-connected layer.

3.2.3. STTras-Block

This block aims to analyze dynamic spatial and temporal features. Its internal structure is shown in Fig. 3. It consists of three modules: STm, TTm and STFm.

STm (dark blue box in Fig. 3) is responsible for dynamic spatial feature extraction. It consists of two main types of layers: Fully-Connected Layer and Spatial Attention Layer. First, the input X_T passes a Fully-Connected layer for embedding more features as $X_T^{STm;fc1} \in \mathbb{R}^{B\times T\times N\times F}$. Then the embedded spatial matrix E_s is joined into $X_T^{STm;fc1}$ by the function, $(X_T^{STm;Es} = X_T^{STm;fc1} + E_s; X_T^{STm;Es} \in \mathbb{R}^{B\times T\times N\times F})$ so as to enable the traffic input to carry spatial dependencies of road segments. To obtain more information, $X_T^{STm;fc1}$ and $X_T^{STm;Es}$ are concatenated as $X_T^{STm;cl} \in \mathbb{R}^{B\times T\times N\times 2F}$ for generating (1) Queries, $Q^S \in \mathbb{R}^{B\times T\times N\times(h\times d_q^s)}$, (2) Keys, $K^S \in \mathbb{R}^{B\times T\times N\times(h\times d_k^s)}$ and (3) Values, $V^S \in \mathbb{R}^{B\times T\times N\times(h\times d_v^s)}$ through three Fully-Connected layers with ReLU activation functions (1) $Q^S = ReLU(w_q^s X_T^{STm;cl})$, (2) $K^S = ReLU(w_k^s X_T^{STm;cl})$ and (3) $V^S = ReLU(w_v^s X_T^{STm;cl})$ where w_q^s, w_k^s and w_v^s are learnable weight matrices, d_q^s, d_k^s and d_v^s are embedded spatial features of each sensor for Q^S, K^S and V^S, respectively and h is the number of heads.

After obtaining the three high dimensional spatially-fused features (Q^S, K^S and V^S), dynamic-spatial dependencies $S^S \in \mathbb{R}^{B\times T\times N\times F}$ are calculated by a Spatial Attention layer, given by

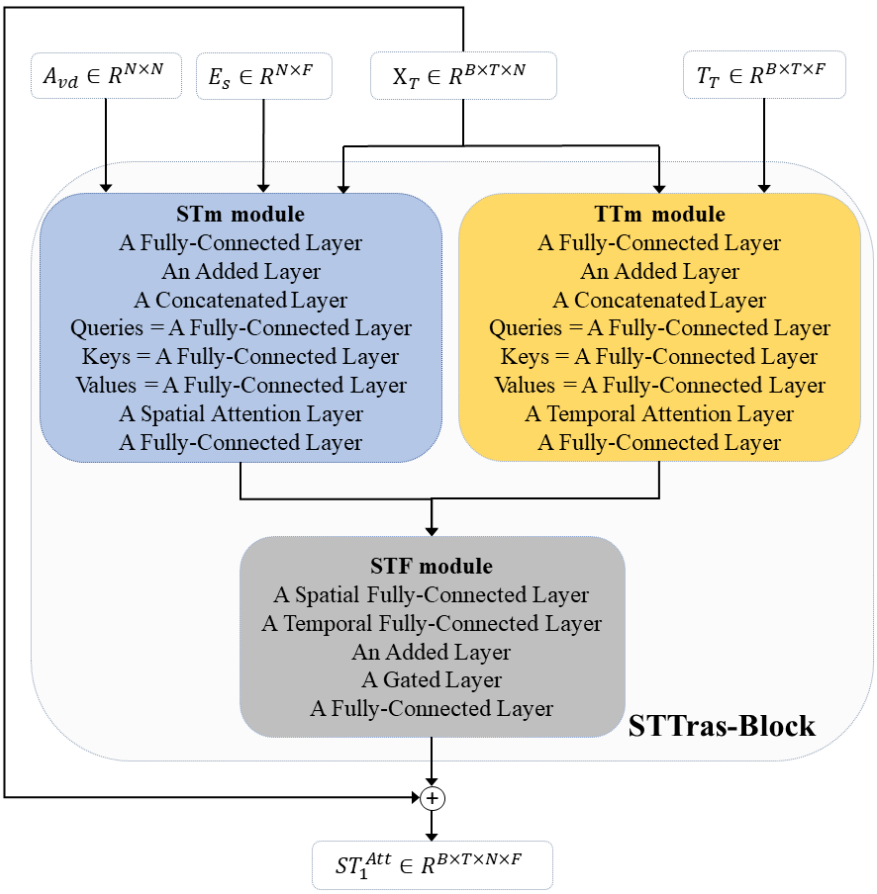


Figure 3: (Color online) The STTras-Block with its constituent modules including (1) STm (the left dark blue box), (2) TTm (the right yellow box) and (3) STF (the bottom grey box)

$$S^S = Softmax\left(\frac{A_{vd}\left(Q^S(K^S)^\intercal\right)}{\sqrt{d_k^s}}\right)V^S \tag{5}$$

where $Q^S(K^S)^\intercal \in \mathbb{R}^{B\times T\times N\times N}$ represents the relationship of each road segment to others in the network and A_{vd} is used to correct this spatial relationship. $\intercal$ stands for matrix transpose operation. Note that the correction of the spatial relationship should be done before multiplying a scaled dot-product attention $\frac{1}{\sqrt{d_k^s}}$. Otherwise, the dot-products grow larger again in magnitude, which could push the Softmax function into regions where it

has extremely small gradients [38]. In addition, A_{vd} is set as "None" in the first STTras-Block to learn the initial physical-spatial dependencies. In order to explore interactions among latent features, a Fully-Connected layer with ReLU activation function is used for computing dynamic spatial features $S^S_{output} \in \mathbb{R}^{B\times T\times N\times F}$ with the residual connection, which is formulated as follow:

$$S^S_{output} = ReLU(w^s_o S^S) + X_T^{STm;fc1} \tag{6}$$

where w^s_o is the learnable weight matrix and S^S_{output} is the output of STm.

TTm (yellow box in Fig. 3) is responsible for dynamic temporal feature extraction. Similar to STm, it consists of Fully-Connected Layer and Temporal Attention Layer. Existing works (e.g., [28, 41, 46]) either used GRU or LSTM to capture temporal features due to their abilities in analyzing long-term dependency using recurrent units to deliver temporal features from the current time interval to the next time interval. However, transformer-based models use the attention mechanism to distribute different weights to traffic data from previous time intervals to contribute to traffic data in the future time intervals.

TTm requires historical traffic data, X_T, and historical-external temporal features, T_T as input. Note that T_T is replaced by the future-external temporal features, $T_{T'}$, in the second STTras-Block. This aims to learn the historical-external temporal features first and then align with the future-external temporal features, $T_{T'}$, to improve prediction. The first Fully-Connected layer embeds more features into X_T as $X_T^{TTm;fc1} \in \mathbb{R}^{B\times T\times N\times F}$, and then an Added layer enables the embedded features to be enhanced by the historical-external temporal features via $X_T^{TTm;T_T} = X_T^{TTm;fc1} + unsqueeze(T_T); X_T^{TTm;T_T} \in \mathbb{R}^{B\times T\times N\times F}$. The $unsqueeze(.)$ function expands an additional dimension to match the $X_T^{TTm;fc1}$ format. To correct the embedded features and obtain more information, $X_T^{TTm;fc1}$ and $X_T^{TTm;T_T}$ are concentrated as $X_T^{TTm;cl} \in \mathbb{R}^{B\times T\times N\times 2F}$ before being sent to three Fully-Connected layers for generating (1) Queries $Q^T \in \mathbb{R}^{B\times N\times T\times (h\times d^t_q)}$, (2) Keys $K^T \in \mathbb{R}^{B\times N\times T\times (h\times d^t_k)}$ and (3) Values $V^T \in \mathbb{R}^{B\times N\times T\times (h\times d^t_v)}$ by Eq. (7).

$$\begin{aligned} Q^T &= ReLU(w^t_q X_T^{TTm;cl}) \\ K^T &= ReLU(w^t_k X_T^{TTm;cl}) \\ V^T &= ReLU(w^t_v X_T^{TTm;cl}) \end{aligned} \tag{7}$$

where w_q^t, w_k^t, w_v^t are learnable weight matrices for Q^T, K^T and V^T, respectively while d_q^t, d_k^t and d_v^t are the corresponding embedded temporal features of each time interval. After achieving the three high dimensional temporally-fused features (Q^T, K^T and V^T), dynamic-temporal dependencies $S^T \in \mathbb{R}^{B\times N\times T\times F}$ are calculated by a Temporal Attention layer as given in Eq. (8).

$$S^T = Softmax\left(\frac{Q^T(K^T)^\intercal}{\sqrt{d_k^t}}\right)V^T \tag{8}$$

where $Q^T(K^T)^\intercal \in \mathbb{R}^{B\times N\times T\times T}$ represents the relationship of each time interval to all others. We follow [47] to set $d_q^t = d_k^t = d_v^t = 8$. Furthermore, a Fully-Connected layer with ReLU activation function is used to generate dynamic temporal features $S_{output}^T \in \mathbb{R}^{B\times N\times T\times F}$ with the residual connection via Eq. (9).

$$S_{output}^T = ReLU(w_o^t S^T) + (X_T^{TTm;fc1})^\intercal \tag{9}$$

where w_o^t is the learnable weight matrix and S_{output}^T is the output of TTm.

STF fuses dynamic-spatial and dynamic-temporal features (S_{output}^S and S_{output}^T) into $S^{ST} \in \mathbb{R}^{B\times T\times N\times F}$ based on the gate mechanism of GRU [43]. First, the dynamic-spatial features S_{output}^S are added to the dynamic-temporal features S_{output}^T after passing a Fully-Connected layer with $Tanh$ activation function (as given in Eq. (10)). Then, the gate mechanism (cf. Eq. (11)) is used for calculating the output of STF, $S_G^{ST} \in \mathbb{R}^{B\times T\times N\times F}$.

$$S^{ST} = Tanh\big(w_{ss}S_{output}^S + w_{st}(S_{output}^T)^\intercal\big) \tag{10}$$

$$S_G^{ST} = S^{ST} \times S_{output}^S + (1 - S^{ST}) \times S_{output}^T \tag{11}$$

where w_{ss} and w_{st} are learnable weight matrices of S_{output}^S and S_{output}^T respectively.

3.2.4. DGCN

This block corrects the learned spatial relationships of sensors or road segments by conducting the convolutional operation on the virtual dynamic spatial matrix A_{vd} and then joining the corrected spatial relationships into S_G^{ST}. It consists of a GCN layer with ReLU activation function and a Batch Normalization layer. The inputs of this block are the updated virtual dynamic

spatial matrix A_{vd} and S_G^{ST}. The output of this block, $GCN_d \in \mathbb{R}^{B\times T\times N\times F}$, representing the corrected spatially-fused features, is computed via Eq. (12).

$$GCN_d = BN(ReLU((w_{dgcn}A_{vd} + b_{dgcn})S_G^{ST})) + S_G^{ST} \tag{12}$$

where $w_{dgcn} \in \mathbb{R}^{N\times N}$ is the learned virtual spatial weight matrix, $b_{dgcn} \in \mathbb{R}^N$ is the related bias and $BN(.)$ is the batch normalization that enhances the stability of the neural network, regularizing the model, and reducing its sensitivity to parameter initialization. $(w_{dgcn}A_{vd} + b_{dgcn})$ is used to capture global structural information while $ReLU(...)$ is ReLU activation function that introduces non-linearity and enables the model to learn complex patterns. In the end, $(... + S_G^{ST})$ represents the residual connection that preserves the original information through the layer, alleviating the problem of vanishing gradients and promoting more effective learning.

3.2.5. Fully-Connected Layer

This fully-connected layer is used for the final prediction. The output from the second STTras-Block for analyzing X_T, $(S_G^{ST})_{2T} \in \mathbb{R}^{B\times T\times N\times F}$, and the output from the second STTras-Block for analyzing X_D, $(S_G^{ST})_{2D} \in \mathbb{R}^{B\times T'\times N\times F_d}$, are concatenated as $S_{TD}^{ST} \in \mathbb{R}^{B\times T\times N\times (F+F_d)}$ (Notes that $T = T'$). The final-connected layer takes the reshaped $S_{TD}^{ST} \in \mathbb{R}^{B\times N\times \left(T\times (F+F_d)\right)}$ as its input and then makes the final prediction (cf Eq. (13)).

$$\tilde{X}_{T+T'} = FC(S_{TD}^{ST}) \tag{13}$$

where FC represents the fully-connected layer.

4. Experiments

4.1. Datasets

To evaluate our VDGCNeT model, two real-world datasets from large-scale road networks, labeled as `PEMS-BAY` and `METR-LA` [26], are used. The locations of loop detectors from both networks are presented in Section 4.5. The `PEMS-BAY` dataset is collected from California Transportation Agencies (CalTrans) Performance Measurement System (PeMS) [48] and contains the traffic speed data from 325 sensor stations in Bay Area. This dataset covers traffic measurements for a period of six months (i.e., 1^{st} January – 30^{th} June

2017). The time interval for the data is 5 minutes and the total number of observed traffic data points is 16,937,700 (= 52,116 × 325). The second real-world dataset, `METR-LA`, is collected from loop detectors in the highways of Los Angeles County [49]. It includes 207 sensors and covers traffic data for a period of four months from the 1^{st} of March to the 30^{th} of June in 2012. The time interval is 5 minutes and the total number of observed traffic data points is 7,094,304 (= $34,272 \times 207$). Compared to `PEMS-BAY`, `METR-LA` exists serious data missing due to sensor incidents. Totally, it missed 575,302 data points, which accounts for 8.11% (= $\frac{575,302}{7,094,304} \times 100\%$) of all data points. TABLE 1 gives the basic statistics of both datasets. From the table, it can be observed that the traffic recorded in `METR-LA` has higher volatility with a larger standard deviation and variance. For both datasets, undirected graphs with edge weights are used to construct their corresponding adjacency matrices. The road distances between sensor locations are first computed and then a thresholded Gaussian Kernel [50] is used to build the adjacency matrix. The edge weights are calculated by Eq. (14):

$$W_{i,j}^{e} = \begin{cases} exp(-\frac{dist(i,j)^2}{2\sigma^2}), & if\ dist(i,j) < d_{threshold} \\ 0, & otherwise \end{cases} \tag{14}$$

where $W_{i,j}^{e}$ is the edge weight between sensor location i and j, and $dist(i,j)$ represents the actual physical distance between sensor location i and j. The standard deviation of the distances is denoted by σ and $d_{threshold}$ is the threshold.

Table 1: Characteristics of both traffic speed datasets

Dataset	Max	Min	Mean	Std	Var	Size
`PEMS-BAY`	85.10	0.00	62.62	8.56	85.41	135.90 MB
`METR-LA`	70.00	0.00	53.72	19.19	374.85	57.00 MB

4.2. Evaluation Metrics

To evaluate our model, we use conventional evaluation metrics as used in [9][16][51]. Specifically, we use three types of errors, including Mean Absolute Error (MAE) (Eq. (15)), Mean Absolute Percentage Error (MAPE) (Eq. (16)) and Root-Mean Square Error (RMSE) (Eq. (17)).

$$MAE = \frac{1}{N \times T'} \cdot \sum_{i=1}^{N} \sum_{t=1}^{T'} |x_t^i - \tilde{x}_t^i| \tag{15}$$

$$MAPE = \frac{1}{N \times T'} \cdot \sum_{i=1}^{N} \sum_{t=1}^{T'} \frac{|x_t^i - \tilde{x}_t^i|}{x_t^i} \times 100\% \tag{16}$$

$$RMSE = \left[\frac{1}{N \times T'} \cdot \sum_{i=1}^{N} \sum_{t=1}^{T'} (x_t^i - \tilde{x}_t^i)^2 \right]^{\frac{1}{2}} \tag{17}$$

MAE is the average absolute difference between the real and predicted traffic states. MAPE is the percentage of absolute difference between the real and predicted traffic states and is utilized to measure the percentage of prediction error. RMSE is the standard deviation of the residuals, which is the difference between the real and predicted traffic states. Finally, we define the accuracy of the prediction as $(100\% - MAPE)$.

4.3. Parameter Study

To optimize VDGCNeT, there are three types of parameters needed to be set and learned as follows:

(1) For input, the parameters include historical time intervals T, targeted time intervals T', previous days D and batch size B. We follow [26][52] and set historical time intervals T, targeted time intervals T', and previous days D as 12, 12 and 1, respectively. For batch size B, we experimentally find the best value as it refers to the number of features $(=B \times T)$ used to update the virtual dynamic spatial matrix A_{vd} and directly affects the performance of our VDGCNeT. If B is too large, it would bring too many features when updating the virtual dynamic spatial matrix. If it is too small, it would not maximize the generalization of the learned virtual dynamic spatial matrix. We show this in Fig. 4 which shows the relationships of batch size and averaged MAE for both `PEMS-BAY` (blue line with dot marker using left y-axis) and `METR-LA` (black line with star marker using right y-axis). For `METR-LA`, the lowest average MAE is achieved when B is equal to 17 while, for `PEMS-BAY`, B is 23. The reason of requiring larger B for `PEMS-BAY` is that `PEMS-BAY` contains 325 sensors and needs more features $(=B \times T)$

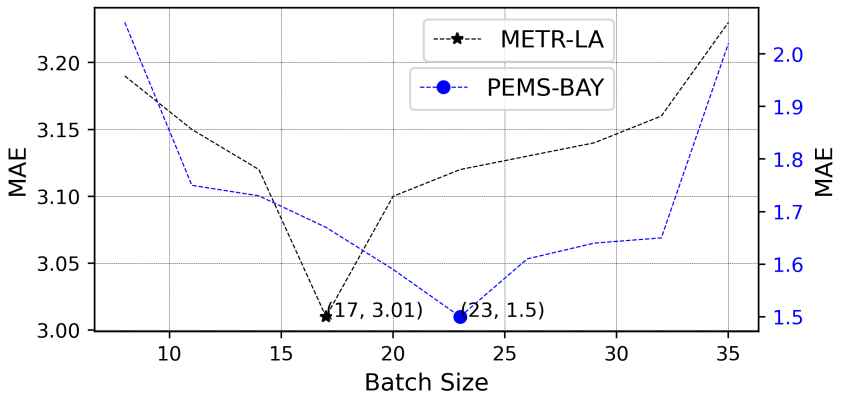


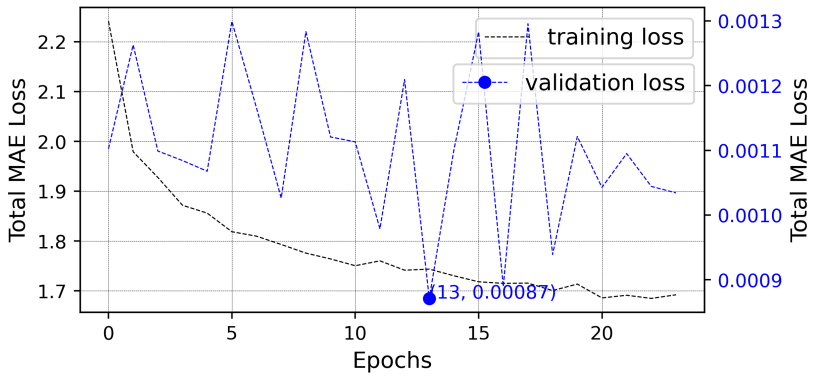
Figure 4: The relationship of batch size and MAE.

to update the virtual dynamic spatial matrix A_{vd}, compared to 207 sensors in `METR-LA`.

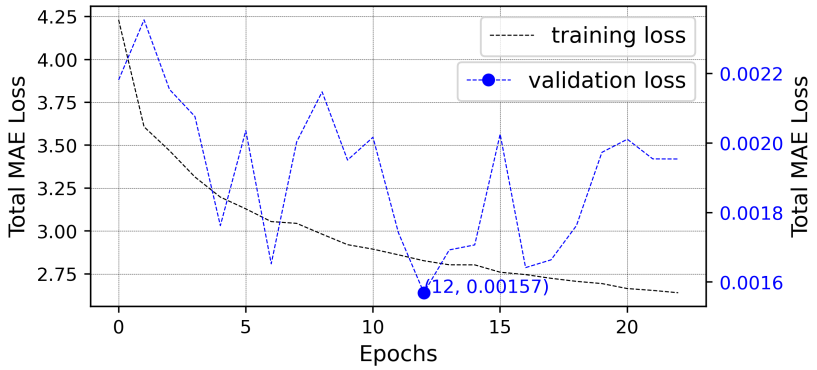
(2) For our model, hyper-parameters include the number of multi-heads h and the number of embedding features $\{d_q^t, d_k^t, d_v^t, d_q^s, d_k^s, d_v^s\}$. We follow [53] to set both the number of embedding features and the number of multi-heads to 8. Therefore, the number of the embedded features F from the first Fully-Connected layer in both STm and TTm is equal to 64 ($= 8 \times 8$). In addition, the total number of parameters for `PEMS-BAY` and `METR-LA` are 298,849 and 235,955, respectively. Again, more parameters are required for `PEMS-BAY` due to the higher number of sensors, resulting in more parameters in DGCN-Block.

(3) For the training phase, parameters include learning rate, r and the number of epochs, e. Slow learning rate will typically make a training algorithm converge slowly and conversely, a large value for learning rate may make the algorithm to diverge. Using experimental methods to find the best learning rate is usually time consuming. In our work, we use the Cyclical Learning Rates (CLR) method [54] to optimize the learning rate. Based on this, the optimized learning rate is $1.12e^{-03}$. In addition, we use *stop early* strategy to find the number of epochs. Specifically, the *stop early* strategy will stop the training process when the training loss continues to decrease in 10 consecutive epochs while the validation loss increases. This avoids the problem of over-fitting. Fig. 5 presents the relationships of training loss, validation loss and the

number of epochs. The x-axis represents the number of epochs and the y-axis gives the MAE loss (left y-axis for training loss and right y-axis for validation loss). Due to the *stop early* strategy, the training process stops at epoch 13 for `PEMS-BAY` (cf. Fig. 5 (a)) and at epoch 12 for `METR-LA` (cf. Fig. 5 (b)) where the validation losses are the lowest.



(a)



(b)

Figure 5: The relationship of training and validation MAE losses with the number of epochs.

Since *Adam* [55] has been shown to be efficient for optimizing parameters of deep learning models, we also use *Adam* as the Optimizer. We follow the convention (e.g., [53]) and use 70% of the dataset for training, 10% for validation and 20% for testing. All experiments are conducted on a machine

equipped with a GeForce RTX 2080 Ti GPU card with 11 GB Memory and 1545MHz Boost Clock and PyTorch is used to implement this work.

4.4. Analysis of the VDGCNeT Model

4.4.1. Comparison against Ground Truth Data

Fig. 6 and Fig. 7 respectively shows the real and predicted traffic states by VDGCNeT and two well-known models in a day from two randomly selected sensors in `PEMS-BAY` and `METR-LA`. The locations of the two selected sensors are shown in Fig. 8 with blue pointers (400030 for `PEMS-BAY` and 773869 for `METR-LA`). In Fig. 7, we indicated serious missing data in the `METR-LA` dataset with shaded area (e.g., traffic speed continues to be Nan between 2012-03-04 08:20:00 and 2012-03-04 09:10:00.) and, even so, VDGCNeT can still predict traffic states and follow the trend over time. Furthermore, our VDGCNeT predicts traffic states more accurate than the other two models, especially for large traffic prediction horizons. Besides, Fig. 6 and Fig. 7 also show that the predictions of VDGCNeT are more accurate on `PEMS-BAY` than on `METR-LA`. From 5-min to 60-min prediction, the predictions are increasingly less accurate but overall, still follow the trend of changes over time.

Fig. 8 compares the real (left) and predicted (right) traffic speed by VDGCNeT on both networks, `PEMS-BAY` in Fig. 8(a) and `METR-LA` in Fig. 8(b). We see close agreements between the real and the predicted traffic speed for both sets of sensors across the two road networks. This suggests that VDGCNeT is capable of offering accurate predictions on large-scale road networks. Besides, Fig. 8 also shows that the very low or high traffic speed happens on several continuous sensors for both road networks. It indicates that traffic states at one location are influenced by its neighbors. Meanwhile, similar traffic speed at a time interval can be observed at sensors which are far away from each other. It indicates that traffic states may be hidden network-wide spatial dependencies. Our model takes such dependency into account by generating a virtual dynamic road graph that describes the hidden and dynamic connections between road segments with respect to traffic states.

4.4.2. Parameter Visualization

In this section, we compare the learned spatial weights computed by VDGCNeT against the physical road network to reveal the hidden spatial dependency. Fig. 9 shows the adjacency matrix A_0 of the physical road network and the learned spatial weights, w_{dgcn} (cf. Eq. (12)), from the DGCN-Block of VDGCNeT model for historical traffic in previous time intervals, X_T and

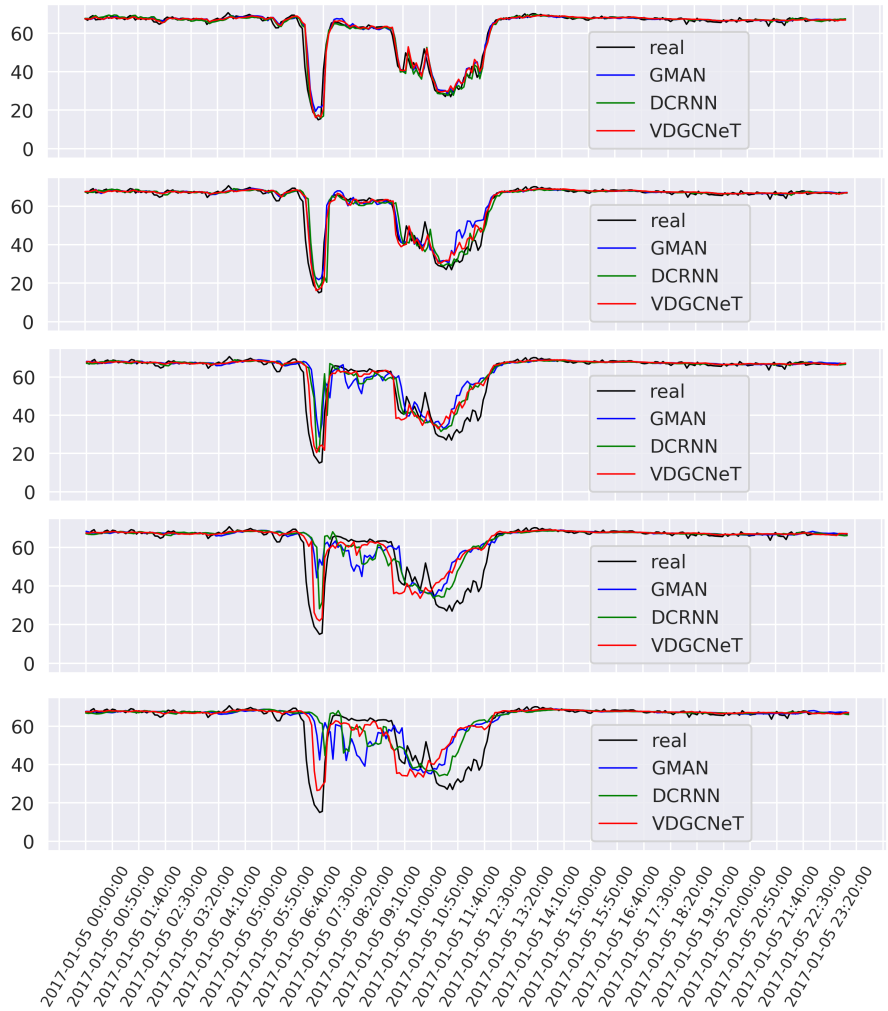


Figure 6: (Color Online) Real (black line) and predicted (red line) traffic speed ($miles/h$) in a week with 288 ($= \frac{1days*24hours*60mins}{5mins}$) time intervals by VDGCNeT on `PEMS-BAY` with a time interval = 5 mins. The x-axis represents the time and the y-axis is traffic speed. The prediction horizons are 5-min, 15-min, 30-min, 45-min and 60-min from top to bottom, respectively.

in the same time interval with the targeted time interval from previous days, X_D on `PEMS-BAY` (top row) and `METR-LA` (bottom row). For clarity, only the first 100 sensors are shown. Both x- and y-axis present the sensor ID. The color describes the spatial dependency relationship (darker = more relevant). By comparing Fig. 9 (b) and (e) against Fig. 9 (c) and (f), we see that the

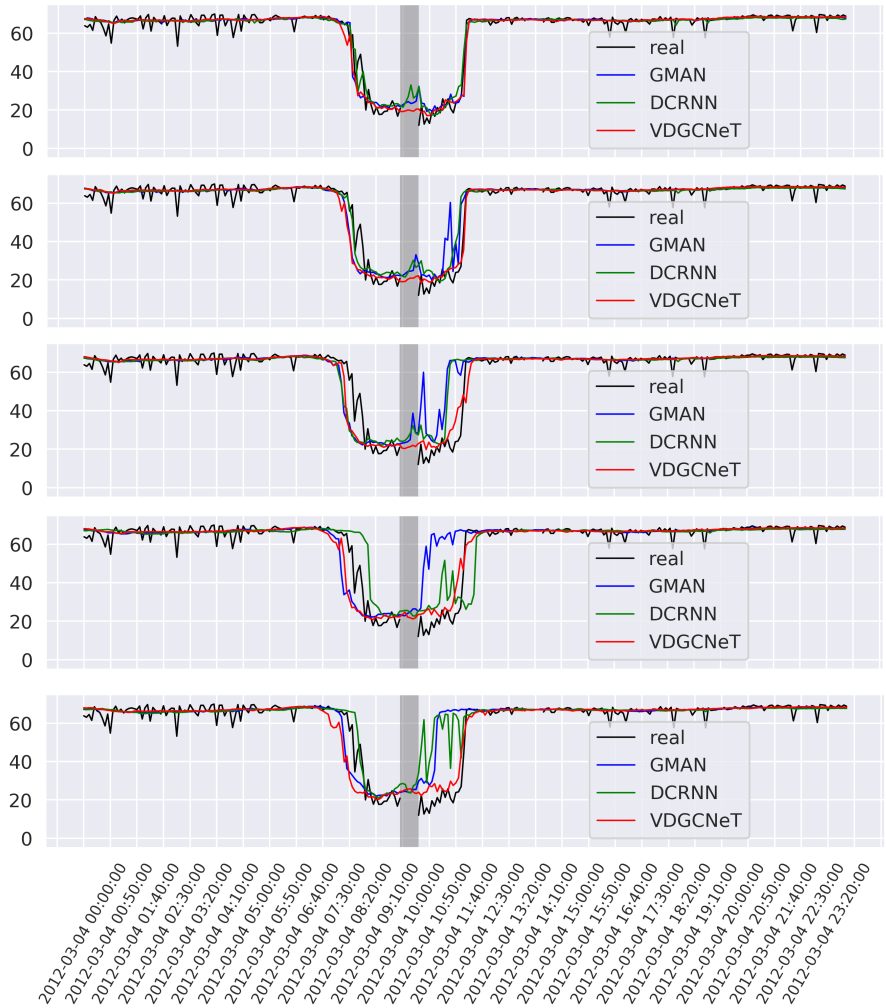
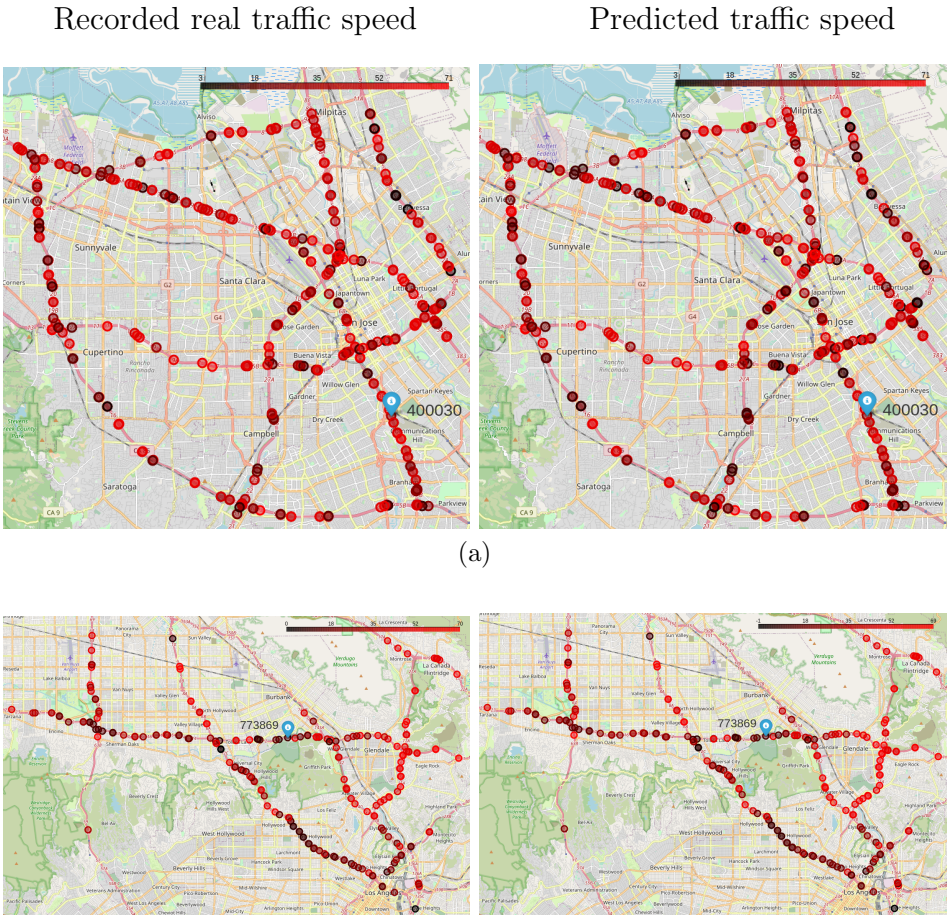


Figure 7: (Color Online) Real (black line) and predicted (red line) traffic speed ($miles/h$) in a week with 288 ($= \frac{1days*24hours*60mins}{5mins}$) time intervals by VDGCNeT on `METR-LA` with a time interval = 5 mins. The x-axis represents the time and the y-axis is traffic speed. The prediction horizons are 5-min, 15-min, 30-min, 45-min and 60-min from top to bottom, respectively.

targeted traffic data is more dependent on T previous time intervals than on the same time interval from D previous days (i.e., darker shades in Fig. 9 (b) and (e)). In addition, the principal diagonal of the spatial weight matrices have significantly darker shade, indicating that traffic states are strongly related to its own historical traffic states. This is more obvious in Fig. 9 (c)

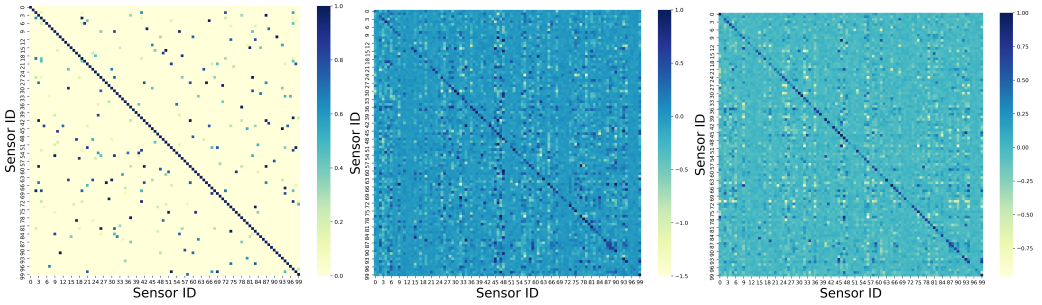


(a)

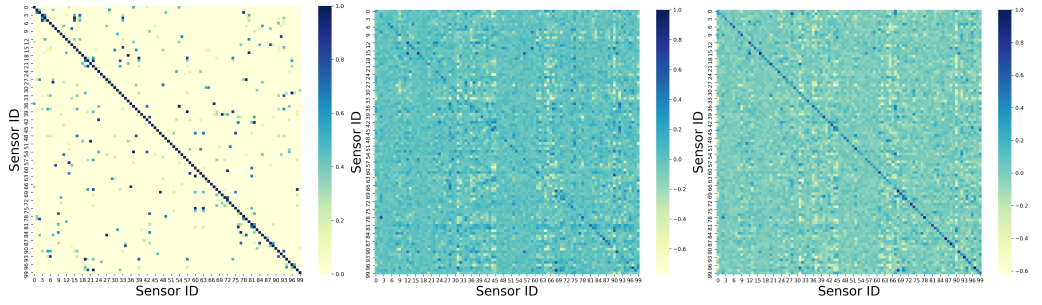
(b)

Figure 8: (Color Online) Visualization of real (left sub-figures) and predicted (right sub-figures) traffic speed at a time interval on two road networks: (a) `PEMS-BAY` and (b) `METR-LA`. The lower traffic speed, the darker color.

and (f) since historical traffic states from neighbors in previous days have less impacts on traffic state in the targeted sensor. Between the two networks, (b) and (c) for `PEMS-BAY` is darker than (e) and (f) for `METR-LA`. It indicates that the spatial dependencies of sensors in `PEMS-BAY` are more significant than `METR-LA`. From spatial weight matrices, some sensors, which are far away from the targeted sensor, still generate important impacts on the targeted sensor. It shows that traffic states of non-adjacent sensors affect each other.



(a) Adjacency matrix A_0 (b) Learned dynamic spatial weights for X_T (c) Learned dynamic spatial weights for X_D



(d) Adjacency matrix A_0 (e) Learned dynamic spatial weights for X_T (f) Learned dynamic spatial weights for X_D

Figure 9: (Color online) The adjacency matrix A_0 and learned dynamic spatial weight matrices for X_T and X_D ({(a), (b), (c)} for `PEMS-BAY` and {(d), (e), (f)} for `METR-LA`). These matrices show the spatial dependencies of the first 100 sensors on both datasets. Both the X-axis and the Y-axis represent the sensor ID. The color indicates the spatial dependency relationship, the darker the more relevant.

4.4.3. Ablation Experiment

We further analyze our VDGCNeT model via ablation experiments to study the contribution and importance of each block towards the final prediction. For this, we create three variants that are built by removing one module from the proposed VDGCNeT as follows.

- **GCNTG**: This variant uses the adjacency matrix A_0 instead of the virtual dynamic spatial matrix A_{vd}. Comparing GCNTG and VDGCNeT allows us to understand the contribution of the virtual dynamic graph proposed in this work to the final prediction.
- **GCNTG***: This variant utilizes the Identity matrix I instead of the

virtual dynamic spatial matrix A_{vd}. Comparison of GCNTG* and VDGCNeT could help us to understand how important of the virtual dynamic graph proposed in this work is to achieve high accuracy of traffic prediction.

- **VTGA**: This variant is developed by removing the DGCN-Block of VDGCNeT and aimed at investigating the role played by the DGCN-Block in the full-fledged VDGCNeT model.

- **VDGCNeT***: For this variant, we remove the second STTras-Block in VDGCNeT to understand the contribution of correcting learned features from DGCN-Block by aligning them with the virtual dynamic spatial matrix A_{vd} in STm and the targeted temporal embedding features $T_{T'}$ in TTm.

TABLE 2 presents the results of our ablation experiments on the two real-world datasets for $T' = \{1, 3, 6, 9, 12\}$ (i.e., {5-min, 15-min, 30-min, 45-min, 60-min} prediction horizons). Overall, VDGCNeT achieves the best performance in both networks. We observe the following:

- The virtual dynamic spatial matrix A_{vd}, generated by the Att-Similar-Block to learn the dynamic spatial relationships of road segments, is important for improving the prediction accuracy. This conclusion is derived from the better result achieved by VDGCNeT over GCNTG and GCNTG*. This further supports our design of VDGCNeT in using virtual dynamic graph rather than on the physical road connectivity.

- The DGCN-Block conducts convolution operation on virtual dynamic graph A_{vd}. It efficiently learns dynamic spatial-temporal features and also joins the spatial relations of historical traffic and future traffic into the learned features from the first STTras-Block. The contribution of the DGCN-block can be seen when comparing the prediction accuracy between the full VDGCNeT and VTGA whereby VTGA performed 0.13% (`PEMS-BAY`) and 0.04% (`METR-LA`) worse without the DGCN-Block.

- The removal of the second STTras-Block results in the largest performance deterioration. This can be observed by the difference in accuracy achieved by the three variants compared to VDGCNeT whereby VGCNTGA* obtained the largest decrease in accuracy. This is due to the

fact that VGCNTGA* loses (1) the functions of the virtual dynamic spatial matrix A_{vd} in STm to correct the learned spatially-fused features and (2) the targeted temporal embedding features $T_{T'}$ in TTm to build the external relationships of historical and future traffic data. Specifically, for the virtual dynamic spatial matrix, A_{vd}, by removing the second STTras-Block, the model can no longer correct the spatial features in the learned spatially-fused features from DGCN-Block.

- The full VDGCNeT model with the special framework design takes advantages of long-term predictions when compared to its three variants. This could be observed from the best results for 30-min, 45-min and 60-min predictions achieved by the full VDGCNeT model.

Table 2: Results from different modules

Model Name	PEMS-BAY			METR-LA		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE
(a) 5-min future prediction *(T'=1)*						
GCNTG	1.1003	2.86	6.8591	**2.3646**	**5.86**	**4.2179**
GCNTG*	0.9910	2.38	3.9845	2.4316	6.21	4.3956
VTGA	0.9674	**1.87**	2.3618	2.3821	5.98	4.2690
VDGCNeT*	1.0249	2.23	**2.0095**	2.4370	6.30	4.3864
VDGCNeT	**0.9400**	1.94	2.5783	2.3828	6.04	4.3205
(b) 15-min future prediction *(T'=3)*						
GCNTG	1.4172	2.94	4.1556	**2.7558**	**7.25**	**5.3458**
GCNTG*	1.3923	3.13	4.0627	2.8219	7.52	5.5491
VTGA	1.2182	2.86	3.3051	2.8954	7.87	5.8241
VDGCNeT*	1.4147	3.09	3.0655	2.8276	7.68	5.5742
VDGCNeT	**1.1678**	**2.38**	**2.3652**	2.7796	7.42	5.4790
(c) 30-min future prediction *(T'=6)*						
GCNTG	1.7085	3.76	4.6548	3.1933	8.96	6.5808
GCNTG*	1.6948	3.75	4.1683	3.1547	8.83	6.4394
VTGA	1.6407	3.48	3.8876	**3.0853**	8.66	6.3775

VDGCNeT*	1.7298	3.91	4.0023	3.1729	9.08	6.5595
VDGCNeT	**1.5648**	**3.35**	**3.5141**	3.0971	**8.62**	**6.3267**
(d) 45-min future prediction *(T'=9)*						
GCNTG	1.8622	4.23	5.1092	3.3315	9.52	6.9529
GCNTG*	1.8569	4.12	4.2767	3.3682	9.68	6.9817
VTGA	1.8513	4.07	4.3450	3.3509	9.80	7.0739
VDGCNeT*	1.8864	4.37	4.3868	3.3884	10.04	7.1581
VDGCNeT	**1.7584**	**3.90**	**4.0445**	**3.3014**	**9.42**	**6.8548**
(e) 60-min future prediction *(T'=12)*						
GCNTG	1.9692	4.55	5.4609	3.4987	10.20	7.3640
GCNTG*	1.9683	4.41	4.5139	3.5237	10.28	7.3468
VTGA	1.9510	4.36	4.5637	3.4580	10.29	7.3303
VDGCNeT*	1.9728	4.61	4.4966	3.5500	10.77	7.5696
VDGCNeT	**1.8769**	**4.27**	**4.3248**	**3.4577**	**10.08**	**7.2180**

4.5. Comparison Study

We proceed to conduct a comparison study pitting our VDGCNeT against the following baseline models.

- GRU [43] is a variant of RNN. It has less gates than LSTM so as to allow faster computing while still retaining competitive performance against LSTM.
- CNN-LSTM [56] integrates CNN and LSTM modules for single-service traffic prediction and interactive network traffic prediction. CNN and LSTM analyze spatial and temporal dependencies, respectively.
- T-GCN [46] uses GCN to learn complex topological structures in the space domain and GRU to learn dynamic changes of traffic data in the time domain.
- TGC-LSTM [28], similar to T-GCN, uses GCN to learn complex topological structures in the space domain and LSTM, instead of GRU, to

learn dynamic changes of traffic data in the time domain. The main difference to T-GCN is extracting spatial features from $k - hop$ neighborhoods and then concatenating them as the final spatial features, instead of only using the features from the $k - hop$ neighborhood for the final prediction.

- STGCN [27] composes of two spatial-temporal convolutional blocks (ST-Conv) and a fully-connected layer. Each ST-Conv block consists of two temporal gated convolution layers and a spatial graph convolution layer in the middle to mine spatial and temporal dependencies.

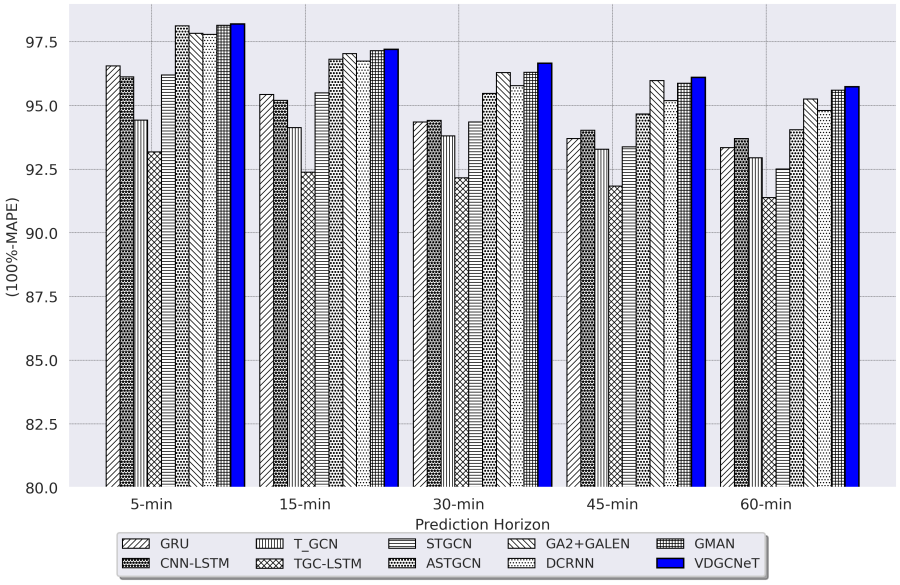
- ASTGCN [52] uses a spatial-temporal attention mechanism to analyze dynamic spatial and temporal features, a GCN for the spatial pattern analysis and a CNN for temporal feature analysis.

- DCRNN [26] models traffic as a diffusion process on a weighted graph using diffusion convolution neural network to learn spatial dependencies and recurrent neural network to learn temporal dependencies.

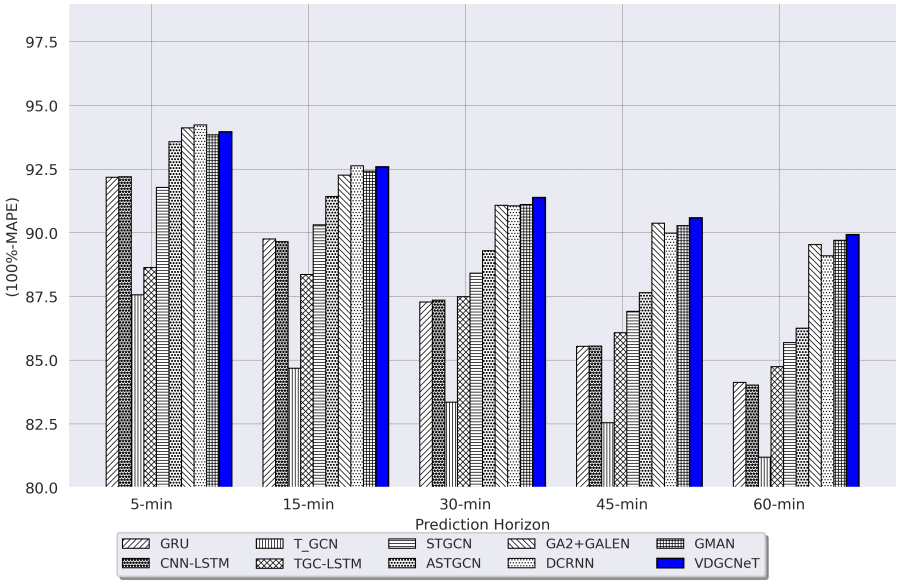
- GMAN [53] models spatial and temporal dependencies in an encoder-decoder architecture. It builds a spatial-temporal embedding to model the graph structure and time information and embeds it into multi-attention mechanisms.

- GA2+GALEN [57] uses a Graph Adjacency LEarning Network (GALEN) to construct traffic graph for a graph learning based traffic predictor of Generative Adversarial Graph attention (GA2). GALEN can capture the inter-sensor correlation from raw historical traffic data using attention on graphs.

Fig. 10 presents the prediction accuracy (100%-MAPE) of all models for both `PEMS-BAY` (Fig. 10 (a)) and `METR-LA` (Fig. 10 (b)). Overall, the prediction accuracy of all models is higher for `PEMS-BAY`. Our VDGCNeT achieves the highest prediction accuracy on average – 96.77% prediction accuracy on average for `PEMS-BAY` and 91.68% for `METR-LA`, followed by GMAN (96.60% and 91.47% respectively), DCRNN (96.05% and 91.39% respectively) and GA2+GALEN (96.47% and 91.47% respectively). At the other end of the spectrum, T-GCN and TGC-LSTM are almost always the worst on both datasets and their average prediction accuracy are 93.71% and 92.18% for



(a) PEMS-BAY



(b) METR-LA

Figure 10: The prediction accuracy (100% - MAPE) of all models for different prediction horizons.

PEMS-BAY and 83.86% and 87.06% for METR-LA, respectively. The one probable reason is that fixed neighbor matrices are used in the GCN module. For instance, spatial features considered by T-GCN are restricted to relations of each sensor with its adjacent neighbors. In addition, the differences of performances among all models become larger over longer prediction horizons. This indicates that all the models including the simpler models (e.g. GRU) compute better predictions for shorter prediction horizons. For longer prediction horizons, more complex models with the abilities to analyze dynamic spatial and temporal dependencies are needed to obtain high prediction accuracy.

TABLE 3 compares the MAE, MAPE and RMSE across both datasets for the nine baseline models and our VDGCNeT. All models perform better for PEMS-BAY than for METR-LA. For instance, their MAPEs are less than 8.00% across all prediction horizons for PEMS-BAY as opposed to under 20.00% for METR-LA. One reason for this is due to the missing data in the METR-LA dataset. From the results, we see that our VDGCNeT mostly obtains the best results for all prediction horizons across both datasets with the exception for 5-min and 15-min prediction horizons where DCRNN narrowly outperforms our model on both datasets. Again, all types of errors achieved by all models increase when the prediction horizons become longer.

Table 3: Results achieved by all models for both datasets

Model Name	PEMS-BAY			METR-LA		
	MAE	MAPE	RMSE	MAE	MAPE	RMSE
(a) 5-min future prediction (T'=1)						
GRU	1.6913	3.45	2.8974	3.3369	7.83	5.8201
CNN-LSTM	1.8765	3.88	3.1958	3.3888	7.80	6.0269
T-GCN	2.4687	5.58	4.5619	4.9543	12.44	8.1131
TGC-LSTM	2.9314	6.83	4.4761	4.6452	11.37	7.5632
STGCN	1.8384	3.81	3.1105	3.6787	8.22	6.5665
ASTGCN	0.9347	1.88	1.7710	2.4857	6.43	4.4114
GA2+GALEN	1.0431	2.17	2.7548	**2.3233**	5.89	4.3219
DCRNN	**0.9170**	2.22	**1.6999**	2.3325	**5.77**	**4.0161**
GMAN	0.9493	1.86	1.7579	2.4445	6.16	4.4177

VDGCNeT	0.9329	**1.81**	1.7064	2.3828	6.04	4.3205
(b) 15-min future prediction (T'=3)						
GRU	2.1303	4.58	3.7916	4.1811	10.25	7.6012
CNN-LSTM	2.1933	4.81	3.9610	4.3127	10.35	7.8845
T-GCN	2.5660	5.88	4.7532	6.1823	15.32	9.7222
TGC-LSTM	3.0190	7.63	4.6935	4.8673	11.64	8.1579
STGCN	2.1103	4.51	3.7309	4.1964	9.69	7.9137
ASTGCN	1.4671	3.19	3.0985	3.0786	8.58	6.0340
GA2+GALEN	1.3610	2.97	3.7819	2.9671	7.74	5.8964
DCRNN	**1.3434**	3.26	**2.8615**	**2.7775**	**7.38**	**5.3535**
GMAN	1.3533	2.86	2.9219	2.8269	7.58	5.6075
VDGCNeT	1.3599	**2.80**	2.9192	2.7796	7.42	5.4790
(c) 30-min future prediction (T'=6)						
GRU	2.5033	5.66	4.5481	5.0948	12.72	9.1599
CNN-LSTM	2.4535	5.59	4.5984	5.1480	12.65	9.3875
T-GCN	2.6843	6.20	5.0002	6.8121	16.65	10.6894
TGC-LSTM	3.0817	7.85	5.0683	5.3021	12.51	9.2798
STGCN	2.4834	5.66	4.6108	4.9270	11.59	9.2209
ASTGCN	1.9311	4.53	4.3212	3.6345	10.71	7.4193
GA2+GALEN	1.6713	3.72	4.7034	3.1367	8.93	6.6617
DCRNN	1.6805	4.23	3.8619	3.1839	8.95	6.4789
GMAN	1.6422	3.71	3.7884	3.1499	8.90	6.5428
VDGCNeT	**1.5648**	**3.35**	**3.5141**	**3.0971**	**8.62**	**6.3267**
(d) 45-min future prediction (T'=9)						
GRU	2.7262	6.31	4.9696	5.6871	14.46	10.0219
CNN-LSTM	2.5918	5.98	4.8798	5.8488	14.45	10.2587
T-GCN	2.8726	6.72	5.3505	7.1436	17.46	11.1565
TGC-LSTM	3.2164	8.17	5.3697	5.7701	13.93	10.3849
STGCN	2.7979	6.63	5.2562	5.5858	13.09	10.1572

ASTGCN	2.2188	5.34	5.0119	4.0540	12.36	8.3200
GA2+GALEN	1.8209	4.03	5.4321	3.3293	9.63	6.9753
DCRNN	1.8706	4.82	4.3794	3.4493	10.02	7.1828
GMAN	1.7885	4.14	4.1667	3.3485	9.72	7.0905
VDGCNeT	**1.7584**	**3.90**	**4.0445**	**3.3014**	**9.42**	**6.8548**
(e) 60-min future prediction (T'=12)						
GRU	2.8705	6.66	5.2104	6.2478	15.88	10.6667
CNN-LSTM	2.6922	6.31	5.0876	6.4159	15.98	10.8960
T-GCN	2.9401	7.06	5.5091	7.7986	18.81	11.9359
TGC-LSTM	3.4623	8.26	5.8917	6.2531	15.26	11.7954
STGCN	3.0762	7.50	5.7680	6.1128	14.32	10.8037
ASTGCN	2.4518	5.96	5.5218	4.4138	13.75	9.0173
GA2+GALEN	1.9041	4.75	6.0316	3.5417	10.47	7.7365
DCRNN	1.9998	5.20	4.6791	3.6755	10.91	7.7280
GMAN	1.8857	4.41	4.3726	3.4815	10.30	7.4331
VDGCNeT	**1.8769**	**4.27**	**4.3248**	**3.4577**	**10.08**	**7.2180**

From TABLE 3, on `PEMS-BAY`, VDGCNeT, GMAN, DCRNN, GA2+GALEN and ASTGCN are always the top five best performers across all prediction horizons. Among them, VDGCNeT always achieves better results as the prediction horizon is increased from 15-min to 60-min. This owes to the virtual dynamic road graph A_{vd} comprehensively mining the hidden spatial dependencies of road segments as well as the spatial- and temporal-transformers in VDGCNeT. For the remaining models, (GRU, CNN-LSTM, T-GCN, TGC-LSTM and STGCN), GRU performs better than others and achieves 3.45% MAPE for 5-min prediction horizon mainly due to its effective function on long-term dependency analysis. TGC-LSTM is the worst performing model with 2.9314, 6.83% and 4.4761 recorded for MAE, MAPE and RMSE respectively. For 30-min, 45-min and 60-min prediction horizons, CNN-LSTM is better than GRU, T-GCN, TGC-LSTM and STGCN. This is likely due to the spatial features extracted by CNN module in CNN-LSTM where it becomes more important for longer prediction horizons. In addition, the 1D

CNN layers in the CNN module enable traffic information from all sensors in the road network to contribute to the targeted sensor, as opposed to T-GCN, TGC-LSTM and STGCN that only consider traffic information from several neighbors. Note that the spatial features extracted by 1D CNN in CNN-LSTM and GCN in T-GCN, TGC-LSTM and STGCN can be considered as global and local spatial features, respectively. To some degree, the global spatial features offer better performance than local spatial features but with high computation cost.

For `METR-LA`, the rank of all considered models is similar to `PEMS-BAY` but the accuracies are worse due to the missing data issue. Similar to `PEMS-BAY`, the top five models are still VDGCNeT, GMAN, DCRNN, GA2+GALEN and ASTGCN. For 5-min and 15-min prediction horizons, DCRNN obtains 5.77% and 7.38% of MAPE followed by our VDGCNeT with 6.04% and 7.42%. For other prediction horizons, VDGCNeT is consistently the best. This is because VDGCNeT not only uses more features from previous time intervals and the same time interval with the targeted interval in previous days to analyze spatial and temporal dependencies, it also generates a dynamic road graph (A_{vd}) to fully mine the hidden and non-uniform spatial relations among sensors or road segments in the network. These can efficiently take into account the sudden changes caused by the missing data and incidents which are considered as big challenges for others. For the other five models (i.e., GRU, CNN-LSTM, T-GCN, TGC-LSTM and STGCN), T-GCN remains the worst performing model with its MAPE increases from 12.44% for 5-min prediction horizon to 18.81% for 60-min prediction horizon. STGCN is almost always the best among these five for all prediction horizons.

4.6. Prediction Accuracy vs Computation Time

We have thus far shown the performance of the different models in terms of prediction accuracy. In this subsection, we turn our attention to the computation time. For this purpose, we run a set of experiments comparing our VDGCNeT with STGCN, ASTGCN, DCRNN and GMAN. To ensure the fairness, for all models, we set the batch size as 18 in the training phase and run them on the same machine equipped with a GeForce RTX 2080 Ti GPU card with 11 GB Memory and 1545MHz Boost Clock. We obtain the average computation time of seven runs. We present the training time of four baselines and our VDGCNeT for both datasets in TABLE 4. From the table, we observe that the training time for both STGCN and ASTGCN are much

shorter than the other models. However, the prediction accuracy of these two models are much lower than the other three, especially when the prediction horizons are large. Among top three models in terms of accuracy (i.e., DCRNN, GMAN and our VDGCNeT), VDGCNeT appears to be the most efficient compared to the other two. The longest training time of DCRNN is caused by the sequence learning in RNN while, for GMAN, the longer training time is due to the setting of the number of ST-Attention Block as $L = 3$ [53] that incur high computation cost. Our experiment results suggest that our proposed model, VDGCNeT, offers the best tradeoff between accuracy and computation time.

Table 4: Computation time of four baselines and our proposed model when the batch size is set as 18.

	Training Time (seconds of per epoch/Total)	
Model Name	`PEMS-BAY`	`METR-LA`
STGCN	196.72s/8458.96s	104.09s/4059.51s
ASTGCN	119.07s/5477.22s	67.64s/3111.44s
DCRNN	662.13s/35087.59s	384.56s/18843.44s
GMAN	834.71s/21702.46s	472.59s/10869.57s
VDGCNeT	516.63s/12036.37s	187.95s/5074.65s

5. Conclusion and Future Works

In this paper, we present a novel deep learning model, VDGCNeT, for addressing the traffic prediction problem on large-scale road networks. Considering the complex and dynamic spatial dependencies of traffic at road segments hidden within the road networks, exploring these hidden and dynamic spatial dependencies is important for achieving high prediction accuracy. Instead of purely relying on the use of the adjacency matrix and other neighborhood matrices that describe the physical connectivity between road segments, we developed an algorithm in the training phase of VDGCNeT to generate a virtual dynamic road graph that comprehensively mines the hidden and dynamic spatial dependency of the road network. We further designed a novel framework for our VDGCNeT based on GCN and the attention mechanism-based Transformers to analyze temporal and spatial dependencies with correction. We trained and tested our VDGCNeT on

two large-scale real-world road networks: PEMS-BAY and METR-LA. We compared it against nine well-known models in literature including: GRU, CNN-LSTM, T-GCN, TGC-LSTM, STGCN, ASTGCN, GA2+GALEN, DCRNN and GMAN. To further gain insights into the characteristic behavior of our model, we conducted ablation experiments and compared the full-fledged VDGCNeT against four of its variants that are built with a removal of one module from the full model. The experimental results indicate that our proposed model, VDGCNeT, obtains the best performance (average accuracy $\approx$ 96.77% for PEMS-BAY and $\approx$ 91.68% for METR-LA) and for almost all prediction horizons, only being closely challenged at the shortest prediction horizon (i.e., at 5-min and 15-min horizons). The MAEs, MAPEs and RMSEs on PEMS-BAY are less than 1.8800, 4.30% and 4.3300, and less than 3.4600, 10.10% and 7.2200 on METR-LA, respectively. These results indicate that our VDGCNeT can efficiently improve the prediction accuracy on large-scale road networks, even on the dataset suffering from missing data (e.g., METR-LA).

Based on findings in this work and other recent related works, we identify three directions that could potentially offer significant improvement of the prediction accuracy in the future.

- For spatial feature analysis, traffic lights around the targeted sensor on the physic road network could be considered as one of features for traffic prediction. This consideration is from observations of very busy traffic around traffic lights. If the intersections on the road network are controlled by traffic lights, time phase for each direction controls traffic flow and also influences traffic nearby. When analyzing traffic on a targeted sensor, traffic lights around this sensor play an important role on the changes of traffic state and could also offer useful spatial features for the traffic prediction.

- For the temporal feature analysis, future traffic state on a road segment heavily depends on previous time steps, particularly under abnormal traffic situation such as serious long-term traffic congestion where traffic state in much more previous time steps are relevant to future traffic state. Furthermore, each road segment in a road network has different traffic situation so that the length of historical traffic data used to predict future traffic states on different road segments should be considered differently. This requires dynamic length of historical traffic data used to predict future traffic state for each sensor along time domain.

- Furthermore, events and weather conditions that influence traffic states by changing peoples' transportation choices could be also considered as features of future traffic prediction.

  For the future work plan, we will focus on the first potential research direction to find the relationship of traffic state on a road segment and time phase of each direction in nearby traffic lights and use this relationship as one of features for future traffic prediction.

References

[1] L. Figueiredo, et al., Towards the development of intelligent transportation systems, in: IEEE 4th Int'l Conf. Intell. Transp. Syst., 2001, pp. 1206–1211.

[2] M. S. Ahmed, A. R. Cook, Analysis of freeway traffic time-series data by using Box-Jenkins techniques, no. 722, 1979.

[3] S. Ho, M. Xie, The use of arima models for reliability forecasting and analysis, Comput. Ind. Eng. 35 (1-2) (1998) 213–216.

[4] M. Van Der Voort, M. Dougherty, et al., Combining kohonen maps with arima time series models to forecast traffic flow, Transp. Res. Part C Emerg. Techol. 4 (5) (1996) 307–318.

[5] B. M. Williams, P. K. Durvasula, D. E. Brown, Urban freeway traffic flow prediction: application of seasonal autoregressive integrated moving average and exponential smoothing models, Transp. Res. Rec. 1644 (1) (1998) 132–141.

[6] M. Lippi, M. Bertini, P. Frasconi, Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning, IEEE Trans. Intell. Transp. Syst. 14 (2) (2013) 871–882.

[7] B. Ghosh, B. Basu, M. O'Mahony, Bayesian time-series model for short-term traffic flow forecasting, J. Transp. Eng. 133 (3) (2007) 180–189.

[8] F. G. Habtemichael, M. Cetin, Short-term traffic flow rate forecasting based on identifying similar traffic patterns, Transp. Res. Part C Emerg. Techol. 66 (2016) 61–78.

[9] W. Huang, et al., Deep architecture for traffic flow prediction: deep belief networks with multitask learning, IEEE Trans. Intell. Transp. Syst. 15 (5) (2014) 2191–2201.

[10] Y. Jia, et al., Traffic speed prediction using deep learning method, in: IEEE Int'l Conf. Intell. Transp. Syst. (ITSC), 2016.

[11] T. Mikolov, et al., Recurrent neural network based language model, in: Annu. Conf. INTERSPEECH., 2010, pp. 1045–1048.

[12] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[13] G. P. Zhang, Time series forecasting using a hybrid arima and neural network model, Neurocomputing 50 (2003) 159–175.

[14] J. Chung, et al., Empirical evaluation of gated recurrent neural networks on sequence modeling, in: NeurIPS Workshop on Deep Learning, 2014.

[15] C. Song, et al., Traffic speed prediction under weekday using convolutional neural networks concepts, in: IEEE Intell. Veh. Symp., 2017, pp. 1293–1298.

[16] Y. Lv, et al., Traffic flow prediction with big data: A deep learning approach., IEEE Trans. Intell. Transp. Syst. 16 (2) (2015) 865–873.

[17] X. Ma, et al., Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction, Sensors 17 (4) (2017) 818.

[18] H. Yu, et al., Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks, Sensors 17 (7) (2017) 1501.

[19] J. Ke, et al., Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach, Transp. Res. Part C Emerg. Techol. 85 (2017) 591–608.

[20] Z. He, et al., Stann: A spatio–temporal attentive neural network for traffic prediction, IEEE Access 7 (2018) 4795–4806.

[21] C. H. Liu, Y. Wang, C. Piao, Z. Dai, Y. Yuan, G. Wang, D. Wu, Time-aware location prediction by convolutional area-of-interest modeling and memory-augmented attentive lstm, IEEE Transactions on Knowledge and Data Engineering 34 (5) (2020) 2472–2484.

[22] G. Zheng, W. K. Chai, V. Katos, An ensemble model for short-term traffic prediction in smart city transportation system, in: IEEE Glob. Commun. Conf. (GLOBECOM), 2019, pp. 1–6.

[23] H. Yao, X. Tang, H. Wei, G. Zheng, Z. Li, Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 33, 2019, pp. 5668–5675.

[24] E. Ahmed, et al., A survey on deep learning advances on different 3d data representations, arXiv preprint arXiv:1808.01462 (2018).

[25] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: 5th Proc. Int'l Conf. Learn. Represent. (ICLR), 2017.

[26] Y. Li, et al., Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, in: Proc. Int'l Conf. Learn. Represent. (ICLR), 2018.

[27] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting, in: IJCAI, 2018.

[28] Z. Cui, et al., Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting, IEEE Trans. Intell. Transp. Syst. 21 (11) (2019) 4883–4894.

[29] Z. Cui, R. Ke, et al., Learning traffic as a graph: A gated graph wavelet recurrent neural network for network-scale traffic prediction, Transp. Res. Part C Emerg. Techol. 115 (2020) 102620.

[30] C. Tian, W. K. Chan, Spatial-temporal attention wavenet: A deep learning framework for traffic prediction considering spatial-temporal dependencies, IET Intell. Transp. Syst. 15 (4) (2021) 549–561.

[31] B. Sun, et al., Modeling global spatial–temporal graph attention network for traffic prediction, IEEE Access 9 (2021) 8581–8594. doi:10.1109/ACCESS.2021.3049556.

[32] G. Zheng, W. K. Chai, V. Katos, A dynamic spatial–temporal deep learning framework for traffic speed prediction on large-scale road networks, Expert Systems with Applications 195 (2022) 116585.

[33] J. Ou, J. Sun, Y. Zhu, H. Jin, Y. Liu, F. Zhang, J. Huang, X. Wang, Stptrellisnets+: Spatial-temporal parallel trellisnets for multi-step metro station passenger flow prediction, IEEE Transactions on Knowledge and Data Engineering (2022) 1–14doi:10.1109/TKDE.2022.3187690.

[34] M. Saberi, et al., A simple contagion process describes spreading of traffic jams in urban networks, Nat. Commun. 11 (1) (2020) 1–9.

[35] H. Peng, et al., Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting, Inf. Sci. 521 (2020) 277–290.

[36] H. Peng, B. Du, et al., Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning, Inf. Sci. 578 (2021) 401–416.

[37] R. Mondal, et al., A new framework for smartphone sensor-based human activity recognition using graph neural network, IEEE Sens. J. 21 (10) (2020) 11461–11468.

[38] A. Vaswani, et al., Attention is all you need, in: 31st Proc. Int'l Conf. NeurIPS, 2017, pp. 6000–6010.

[39] P. J. Bickel, et al., Measuring traffic, Stat. Sci. 22 (4) (2007) 581–597.

[40] Z. Cui, et al., Graph markov network for traffic forecasting with missing data, Transp. Res. Part C Emerg. Techol. 117 (2020) 102671.

[41] Z. Zhang, et al., Multistep speed prediction on traffic networks: A deep learning approach considering spatio-temporal dependencies, Transp. Res. Part C Emerg. Techol. 105 (2019) 297–322.

[42] X. Shi, et al., A spatial-temporal attention approach for traffic prediction, IEEE Trans. Intell. Transp. Syst. (2020) 1–10.

[43] R. Fu, et al., Using lstm and gru neural network methods for traffic flow prediction, in: 31st Youth Acad. Annu. Conf. Chin. Assoc. Autom., IEEE, 2016, pp. 324–328.

[44] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: 22nd ACM SIGKDD, 2016, pp. 855–864.

[45] R. Karthiga, et al., Transfer learning based breast cancer classification using one-hot encoding technique, in: Int'l Conf. on Artificial Intelligence & Smart Syst. (ICAIS), IEEE, 2021.

[46] L. Zhao, et al., T-gcn: A temporal graph convolutional network for traffic prediction, IEEE Trans. Intell. Transp. Syst. 21 (9) (2019) 3848–3858.

[47] M. Xu, et al., Spatial-temporal transformer networks for traffic flow forecasting, arXiv preprint arXiv:2001.02908 (2020).

[48] C. Chen, Freeway performance measurement system (pems), Public Roads 57 (3) (1994) 8–14.

[49] H. Jagadish, et al., Big data and its technical challenges, Commun. ACM 57 (7) (2014) 86–94.

[50] D. I. Shuman, et al., The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, IEEE Signal Process. Mag. 30 (3) (2013) 83–98.

[51] M.-C. Tan, et al., An aggregation approach to short-term traffic flow prediction, IEEE Trans. Intell. Transp. Syst. 10 (1) (2009) 60–69.

[52] S. Guo, et al., Attention based spatial-temporal graph convolutional networks for traffic flow forecasting, in: AAAI, Vol. 33, 2019, pp. 922–929.

[53] C. Zheng, et al., Gman: A graph multi-attention network for traffic prediction, in: AAAI, Vol. 34, 2020, pp. 1234–1241.

[54] L. N. Smith, Cyclical learning rates for training neural networks, in: IEEE Winter Conf. Appl. Comput. Vis., 2017, pp. 464–472.

[55] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: 3th Proc. Int'l Conf. Learn. Represent. (ICLR), 2014.

[56] X. Cao, et al., Interactive temporal recurrent convolution network for traffic prediction in data centers, IEEE Access 6 (2017) 5276–5289.

[57] J. James, Graph construction for traffic prediction: a data-driven approach, IEEE Transactions on Intelligent Transportation Systems 23 (9) (2022) 15015–15027.