Graph WaveNet for Deep Spatial–Temporal Graph Modeling

Preprint · May 2019

5 authors, including:

Zonghan Wu
University of Technology Sydney
8 PUBLICATIONS   **632** CITATIONS

Shirui Pan
Monash University (Australia)
140 PUBLICATIONS   **2,760** CITATIONS

Guodong Long
University of Technology Sydney
124 PUBLICATIONS   **2,330** CITATIONS

Jing Jiang
University of Technology Sydney
60 PUBLICATIONS   **655** CITATIONS

Some of the authors of this publication are also working on these related projects:

Project — Cohort Discovery and Activity Mining for Policy Impact Prediction View project

Project — Network Embedding and Graph Neural Networks View project

Graph WaveNet for Deep Spatial-Temporal Graph Modeling

Zonghan Wu[1] , **Shirui Pan**[2]* , **Guodong Long**[1] , **Jing Jiang**[1] , **Chengqi Zhang**[1]

[1]Centre for Artificial Intelligence, FEIT, University of Technology Sydney, Australia
[2]Faculty of Information Technology, Monash University, Australia

zonghan.wu-3@student.uts.edu.au, shirui.pan@monash.edu,
{guodong.long, jing.jiang, chengqi.zhang}@uts.edu.au

Abstract

Spatial-temporal graph modeling is an important task to analyze the spatial relations and temporal trends of components in a system. Existing approaches mostly capture the spatial dependency on a fixed graph structure, assuming that the underlying relation between entities is pre-determined. However, the explicit graph structure (relation) does not necessarily reflect the true dependency and genuine relation may be missing due to the incomplete connections in the data. Furthermore, existing methods are ineffective to capture the temporal trends as the RNNs or CNNs employed in these methods cannot capture long-range temporal sequences. To overcome these limitations, we propose in this paper a novel graph neural network architecture, Graph WaveNet, for spatial-temporal graph modeling. By developing a novel adaptive dependency matrix and learn it through node embedding, our model can precisely capture the hidden spatial dependency in the data. With a stacked dilated 1D convolution component whose receptive field grows exponentially as the number of layers increases, Graph WaveNet is able to handle very long sequences. These two components are integrated seamlessly in a unified framework and the whole framework is learned in an end-to-end manner. Experimental results on two public traffic network datasets, METR-LA and PEMS-BAY, demonstrate the superior performance of our algorithm.

1 Introduction

Spatial-temporal graph modeling has received increasing attention with the advance of graph neural networks. It aims to model the dynamic node-level inputs by assuming inter-dependency between connected nodes, as demonstrated by Figure 1. Spatial-temporal graph modeling has wide applications in solving complex system problems such as traffic speed forecasting [Li *et al.*, 2018b], taxi demand prediction [Yao *et al.*, 2018], human action recognition [Yan
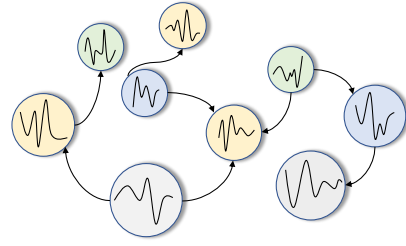
---
*Corresponding Author.

Figure 1: Spatial-temporal graph modeling. In a spatial-temporal graph, each node has dynamic input features. The aim is to model each node's dynamic features given the graph structure.

et al., 2018], and driver maneuver anticipation [Jain *et al.*, 2016]. For a concrete example, in traffic speed forecasting, speed sensors on roads of a city form a graph where the edge weights are judged by two nodes' Euclidean distance. As the traffic congestion on one road could cause lower traffic speed on its incoming roads, it is natural to consider the underlying graph structure of the traffic system as the prior knowledge of inter-dependency relationships among nodes when modeling time series data of the traffic speed on each road.

A basic assumption behind spatial-temporal graph modeling is that a node's future information is conditioned on its historical information as well as its neighbors' historical information. Therefore how to capture spatial and temporal dependencies simultaneously becomes a primary challenge. Recent studies on spatial-temporal graph modeling mainly follow two directions. They either integrate graph convolution networks (GCN) into recurrent neural networks (RNN) [Seo *et al.*, 2018; Li *et al.*, 2018b] or into convolution neural networks (CNN) [Yu *et al.*, 2018; Yan *et al.*, 2018]. While having shown the effectiveness of introducing the graph structure of data into a model, these approaches face two major shortcomings.

First, these studies assume the graph structure of data reflects the genuine dependency relationships among nodes. However, there are circumstances when a connection does not entail the inter-dependency relationship between two nodes and when the inter-dependency relationship between two nodes exists but a connection is missing. To give each circumstance an example, let us consider a recommendation system. In the first case, two users are connected, but they may have distinct preferences over products. In the second case, two

users may share a similar preference, but they are not linked together. Zhang *et al.* [2018] used attention mechanisms to address the first circumstance by adjusting the dependency weight between two connected nodes, but they failed to consider the second circumstance.

Second, current studies for spatial-temporal graph modeling are ineffective to learn temporal dependencies. RNN-based approaches suffer from time-consuming iterative propagation and gradient explosion/vanishing for capturing long-range sequences [Seo *et al.*, 2018; Li *et al.*, 2018b; Zhang *et al.*, 2018]. On the contrary, CNN-based approaches enjoy the advantages of parallel computing, stable gradients and low memory requirement [Yu *et al.*, 2018; Yan *et al.*, 2018]. However, these works need to use many layers in order to capture very long sequences because they adopt standard 1D convolution whose receptive field size grows linearly with an increase in the number of hidden layers.

In this work, we present a CNN-based method named Graph WaveNet, which addresses the two shortcomings we have aforementioned. We propose a graph convolution layer in which a self-adaptive adjacency matrix can be learned from the data through an end-to-end supervised training. In this way, the self-adaptive adjacency matrix preserves hidden spatial dependencies. Motivated by WaveNet [Oord *et al.*, 2016], we adopt stacked dilated casual convolutions to capture temporal dependencies. The receptive field size of stacked dilated casual convolution networks grows exponentially with an increase in the number of hidden layers. With the support of stacked dilated casual convolutions, Graph WaveNet is able to handle spatial-temporal graph data with long-range temporal sequences efficiently and effectively. The main contributions of this work are as follows:

- We construct a self-adaptive adjacency matrix which preserves hidden spatial dependencies. Our proposed self-adaptive adjacency matrix is able to uncover unseen graph structures automatically from the data without any guidance of prior knowledge. Experiments validate that our method improves the results when spatial dependencies are known to exist but are not provided.

- We present an effective and efficient framework to capture spatial-temporal dependencies simultaneously. The core idea is to assemble our proposed graph convolution with dilated casual convolution in a way that each graph convolution layer tackles spatial dependencies of nodes' information extracted by dilated casual convolution layers at different granular levels.

- We evaluate our proposed model on traffic datasets and achieve state-of-the-art results with low computation costs. The source codes of Graph WaveNet are publicly available from `https://github.com/nnzhan/Graph-WaveNet`.

2 Related Works

2.1 Graph Convolution Networks

Graph convolution networks are building blocks for learning graph-structured data [Wu *et al.*, 2019]. They are widely applied in domains such as node embedding [Pan *et al.*, 2018], node classification [Kipf and Welling, 2017], graph classification [Ying *et al.*, 2018], link prediction [Zhang and Chen, 2018] and node clustering [Wang *et al.*, 2017]. There are two mainstreams of graph convolution networks, the spectral-based approaches and the spatial-based approaches. Spectral-based approaches smooth a node's input signals using graph spectral filters [Bruna *et al.*, 2014; Defferrard *et al.*, 2016; Kipf and Welling, 2017]. Spatial-based approaches extract a node's high-level representation by aggregating feature information from neighborhoods [Atwood and Towsley, 2016; Gilmer *et al.*, 2017; Hamilton *et al.*, 2017]. In these approaches, the adjacency matrix is considered as prior knowledge and is fixed throughout training. Monti *et al.* [2017] learned the weight of a node's neighbor through Gaussian kernels. Velickovic *et al.* [2017] updated the weight of a node's neighbor via attention mechanisms. Liu *et al.* [2019] proposed an adaptive path layer to explore the breadth and depth of a node's neighborhood. Although these methods assume the contribution of each neighbor to the central node is different and need to be learned, they still rely on a predefined graph structure. Li *et al.* [2018a] adopted distance metrics to adaptively learn a graph's adjacency matrix for graph classification problems. This generated adjacency matrix is conditioned on nodes' inputs. As inputs of a spatial-temporal graph are dynamic, their method is unstable for spatial-temporal graph modeling.

2.2 Spatial-temporal Graph Networks

The majority of Spatial-temporal Graph Networks follows two directions, namely, RNN-based and CNN-based approaches. One of the early RNN-based methods captured spatial-temporal dependencies by filtering inputs and hidden states passed to a recurrent unit using graph convolution [Seo *et al.*, 2018]. Later works adopted different strategies such as diffusion convolution [Li *et al.*, 2018b] and attention mechanisms [Zhang *et al.*, 2018] to improve model performance. Another parallel work used node-level RNNs and edge-level RNNs to handle different aspects of temporal information [Jain *et al.*, 2016]. The main drawbacks of RNN-based approaches are that it becomes inefficient for long sequences and its gradients are more likely to explode when they are combined with graph convolution networks. CNN-based approaches combine a graph convolution with a standard 1D convolution [Yu *et al.*, 2018; Yan *et al.*, 2018]. While being computationally efficient, these two approaches have to stack many layers or use global pooling to expand the receptive field of a neural network model.

3 Methodology

In this section, we first give the mathematical definition of the problem we are addressing in this paper. Next, we describe two building blocks of our framework, the graph convolution layer (GCN) and the temporal convolution layer (TCN). They work together to capture the spatial-temporal dependencies. Finally, we outline the architecture of our framework.

3.1 Problem Definition

A graph is represented by $G = (V, E)$ where V is the set of nodes and E is the set of edges. The adjacency matrix derived from a graph is denoted by $\mathbf{A} \in \mathbf{R}^{N \times N}$. If $v_i, v_j \in V$ and $(v_i, v_j) \in E$, then $\mathbf{A}_{ij}$ is one otherwise it is zero. At each time step t, the graph G has a dynamic feature matrix $\mathbf{X}^{(t)} \in \mathbf{R}^{N \times D}$. In this paper, the feature matrix is used interchangeably with graph signals. Given a graph G and its historical S step graph signals, our problem is to learn a function f which is able to forecast its next T step graph signals. The mapping relation is represented as follows

$$[\mathbf{X}^{(t-S):t}, G] \xrightarrow{f} \mathbf{X}^{(t+1):(t+T)}, \qquad (1)$$

where $\mathbf{X}^{(t-S):t} \in \mathbf{R}^{N \times D \times S}$ and $\mathbf{X}^{(t+1):(t+T)} \in \mathbf{R}^{N \times D \times T}$.

3.2 Graph Convolution Layer

Graph convolution is an essential operation to extract a node's features given its structural information. Kipf *et al.* [2017] proposed a first approximation of Chebyshev spectral filter [Defferrard *et al.*, 2016]. From a spatial-based perspective, it smoothed a node's signal by aggregating and transforming its neighborhood information. The advantages of their method are that it is a compositional layer, its filter is localized in space, and it supports multi-dimensional inputs. Let $\tilde{\mathbf{A}} \in \mathbf{R}^{N \times N}$ denote the normalized adjacency matrix with self-loops, $\mathbf{X} \in \mathbf{R}^{N \times D}$ denote the input signals , $\mathbf{Z} \in \mathbf{R}^{N \times M}$ denote the output, and $\mathbf{W} \in \mathbf{R}^{D \times M}$ denote the model parameter matrix, in [Kipf and Welling, 2017] the graph convolution layer is defined as

$$\mathbf{Z} = \tilde{\mathbf{A}} \mathbf{X} \mathbf{W}. \qquad (2)$$

Li *et al.* [2018b] proposed a diffusion convolution layer which proves to be effective in spatial-temporal modeling. They modeled the diffusion process of graph signals with K finite steps. We generalize its diffusion convolution layer into the form of Equation 2, which results in,

$$\mathbf{Z} = \sum_{k=0}^{K} \mathbf{P}^k \mathbf{X} \mathbf{W_k}, \qquad (3)$$

where $\mathbf{P}^k$ represents the power series of the transition matrix. In the case of an undirected graph, $\mathbf{P} = \mathbf{A}/rowsum(\mathbf{A})$. In the case of a directed graph, the diffusion process have two directions, the forward and backward directions, where the forward transition matrix $\mathbf{P}_f = \mathbf{A}/rowsum(\mathbf{A})$ and the backward transition matrix $\mathbf{P}_b = \mathbf{A^T}/rowsum(\mathbf{A^T})$. With the forward and the backward transition matrix, the diffusion graph convolution layer is written as

$$\mathbf{Z} = \sum_{k=0}^{K} \mathbf{P}_f^k \mathbf{X} \mathbf{W}_{k1} + \mathbf{P}_b^k \mathbf{X} \mathbf{W}_{k2}. \qquad (4)$$

Self-adaptive Adjacency Matrix: In our work, we propose a self-adaptive adjacency matrix $\tilde{\mathbf{A}}_{adp}$. This self-adaptive adjacency matrix does not require any prior knowledge and is learned end-to-end through stochastic gradient descent. In

doing so, we let the model discover hidden spatial dependencies by itself. We achieve this by randomly initializing two node embedding dictionaries with learnable parameters $\mathbf{E}_1, \mathbf{E}_2 \in \mathbf{R}^{N \times c}$. We propose the self-adaptive adjacency matrix as

$$\tilde{\mathbf{A}}_{adp} = SoftMax(ReLU(\mathbf{E}_1 \mathbf{E}_2^T)). \qquad (5)$$

We name $\mathbf{E1}$ as the source node embedding and $\mathbf{E2}$ as the target node embedding. By multiplying $\mathbf{E1}$ and $\mathbf{E2}$, we derive the spatial dependency weights between the source nodes and the target nodes. We use the ReLU activation function to eliminate weak connections. The SoftMax function is applied to normalize the self-adaptive adjacency matrix. The normalized self-adaptive adjacency matrix, therefore, can be considered as the transition matrix of a hidden diffusion process. By combining pre-defined spatial dependencies and self-learned hidden graph dependencies, we propose the following graph convolution layer

$$\mathbf{Z} = \sum_{k=0}^{K} \mathbf{P}_f^k \mathbf{X} \mathbf{W}_{k1} + \mathbf{P}_b^k \mathbf{X} \mathbf{W}_{k2} + \tilde{\mathbf{A}}_{apt}^k \mathbf{X} \mathbf{W}_{k3}. \qquad (6)$$

When the graph structure is unavailable, we propose to use the self-adaptive adjacency matrix alone to capture hidden spatial dependencies, i.e.,

$$\mathbf{Z} = \sum_{k=0}^{K} \tilde{\mathbf{A}}_{apt}^k \mathbf{X} \mathbf{W}_k. \qquad (7)$$

It is worth to note that our graph convolution falls into spatial-based approaches. Although we use graph signals interchangeably with node feature matrix for consistency, our graph convolution in Equation 7 indeed is interpreted as aggregating transformed feature information from different orders of neighborhoods.

3.3 Temporal Convolution Layer

We adopt the dilated causal convolution [Yu and Koltun, 2016] as our temporal convolution layer (TCN) to capture a node's temporal trends. Dilated causal convolution networks allow an exponentially large receptive field by increasing the layer depth. As opposed to RNN-based approaches, dilated casual convolution networks are able to handle long-range sequences properly in a non-recursive manner, which facilitates parallel computation and alleviates the gradient explosion problem. The dilated causal convolution preserves the temporal causal order by padding zeros to the inputs so that predictions made on the current time step only involve historical information. As a special case of standard 1D-convolution, the dilated causal convolution operation slides over inputs by skipping values with a certain step, as illustrated by Figure 2. Mathematically, given a 1D sequence input $\mathbf{x} \in \mathbf{R}^T$ and a filter $\mathbf{f} \in \mathbf{R}^K$, the dilated causal convolution operation of $\mathbf{x}$ with $\mathbf{f}$ at step t is represented as

$$\mathbf{x} \star \mathbf{f}(t) = \sum_{s=0}^{K-1} \mathbf{f}(s) \mathbf{x}(t - d \times s), \qquad (8)$$

where d is the dilation factor which controls the skipping distance. By stacking dilated causal convolution layers with dilation factors in an increasing order, the receptive field of a
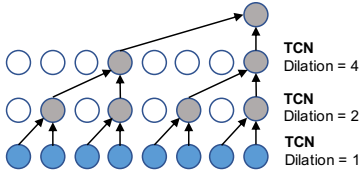
Figure 2: Dilated casual convolution with kernel size 2. With a dilation factor k, it picks inputs every k step and applies the standard 1D convolution to the selected inputs.

model grows exponentially. It enables dilated causal convolution networks to capture longer sequences with less layers, which saves computation resources.

Gated TCN: Gating mechanisms are critical in recurrent neural networks. They have been shown to be powerful to control information flow through layers for temporal convolution networks as well [Dauphin *et al.*, 2017]. A simple Gated TCN only contains an output gate. Given the input $\mathcal{X} \in R^{N \times D \times S}$, it takes the form

$$\mathbf{h} = g(\mathbf{\Theta_1} \star \mathcal{X} + \mathbf{b}) \odot \sigma(\mathbf{\Theta_2} \star \mathcal{X} + \mathbf{c}), \quad (9)$$

where $\mathbf{\Theta_1}$, $\mathbf{\Theta_2}$, $\mathbf{b}$ and $\mathbf{c}$ are model parameters, $\odot$ is the element-wise product, $g(\cdot)$ is an activation function of the outputs, and $\sigma(\cdot)$ is the sigmoid function which determines the ratio of information passed to the next layer. We adopt Gated TCN in our model to learn complex temporal dependencies. Although we empirically set the tangent hyperbolic function as the activation function $g(\cdot)$, other forms of Gated TCN can be easily fitted into our framework, such as an LSTM-like Gated TCN [Kalchbrenner *et al.*, 2016].

3.4 Framework of Graph WaveNet

We present the framework of Graph WaveNet in Figure 3. It consists of stacked spatial-temporal layers and an output layer. A spatial-temporal layer is constructed by a graph convolution layer (GCN) and a gated temporal convolution layer (Gated TCN) which consists of two parallel temporal convolution layers (TCN-a and TCN-b). By stacking multiple spatial-temporal layers, Graph WaveNet is able to handle spatial dependencies at different temporal levels. For example, at the bottom layer, GCN receives short-term temporal information while at the top layer GCN tackles long-term temporal information. The inputs $\mathbf{h}$ to a graph convolution layer in practice are three-dimension tensors with size [N,C,L] where N is the number of nodes, and C is the hidden dimension, L is the sequence length. We apply the graph convolution layer to each of $\mathbf{h}[:, :, i] \in \mathbf{R}^{N \times C}$.

We choose to use mean absolute error (MAE) as the training objective of Graph WaveNet, which is defined by

$$L(\hat{\mathbf{X}}^{(t+1):(t+T)}; \mathbf{\Theta}) = \frac{1}{TND} \sum_{i=1}^{i=T} \sum_{j=1}^{j=N} \sum_{k=1}^{k=D} |\hat{\mathbf{X}}_{jk}^{(t+i)} - \mathbf{X}_{jk}^{(t+i)}| \quad (10)$$

Unlike previous works such as [Li *et al.*, 2018b; Yu *et al.*, 2018], our Graph WaveNet outputs $\hat{\mathbf{X}}^{(t+1):(t+T)}$ as a whole rather than generating $\hat{\mathbf{X}}^{(t)}$ recursively through T steps. It addresses the problem of inconsistency between training and
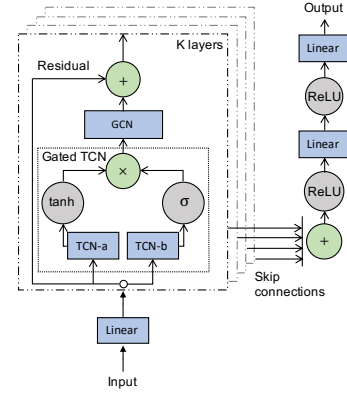

Figure 3: The framework of Graph WaveNet. It consists of K spatial-temporal layers on the left and an output layer on the right. The inputs are first transformed by a linear layer and then passed to the gated temporal convolution module (Gated TCN) followed by the graph convolution layer (GCN). Each spatial-temporal layer has residual connections and is skip-connected to the output layer.

Data	#Nodes	#Edges	#Time Steps
METR-LA	207	1515	34272
PEMS-BAY	325	2369	52116

Table 1: Summary statistics of METR-LA and PEMS-BAY.

testing due to the fact that a model learns to make predictions for one step during training and is expected to produce predictions for multiple steps during inference. To achieve this, we artificially design the receptive field size of Graph WaveNet equals to the sequence length of the inputs so that in the last spatial-temporal layer the temporal dimension of the outputs exactly equals to one. After that we set the number of output channels of the last layer as a factor of step length T to get our desired output dimension.

4 Experiments

We verify Graph WaveNet on two public traffic network datasets, METR-LA and PEMS-BAY released by Li *et al.* [2018b]. METR-LA records four months of statistics on traffic speed on 207 sensors on the highways of Los Angeles County. PEMS-BAY contains six months of traffic speed information on 325 sensors in the Bay area. We adopt the same data pre-processing procedures as in [Li *et al.*, 2018b]. The readings of the sensors are aggregated into 5-minutes windows. The adjacency matrix of the nodes is constructed by road network distance with a thresholded Gaussian kernel [Shuman *et al.*, 2012]. Z-score normalization is applied to inputs. The datasets are split in chronological order with 70% for training, 10% for validation and 20% for testing. Detailed dataset statistics are provided in Table 1.

4.1 Baselines

We compare Graph WaveNet with the following models.

- ARIMA. Auto-Regressive Integrated Moving Average model with Kalman filter [Li *et al.*, 2018b].

Data	Models	15 min			30 min			60 min		
		MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
METR-LA	ARIMA [Li *et al.*, 2018b]	3.99	8.21	9.60%	5.15	10.45	12.70%	6.90	13.23	17.40%
	FC-LSTM [Li *et al.*, 2018b]	3.44	6.30	9.60%	3.77	7.23	10.90%	4.37	8.69	13.20%
	WaveNet [Oord *et al.*, 2016]	2.99	5.89	8.04%	3.59	7.28	10.25%	4.45	8.93	13.62%
	DCRNN [Li *et al.*, 2018b]	2.77	5.38	7.30%	3.15	6.45	8.80%	3.60	7.60	10.50%
	GGRU [Zhang *et al.*, 2018]	2.71	5.24	6.99%	3.12	6.36	8.56%	3.64	7.65	10.62%
	STGCN [Yu *et al.*, 2018]	2.88	5.74	7.62%	3.47	7.24	9.57%	4.59	9.40	12.70%
	Graph WaveNet	**2.69**	**5.15**	**6.90**%	**3.07**	**6.22**	**8.37**%	**3.53**	**7.37**	**10.01**%
PEMS-BAY	ARIMA [Li *et al.*, 2018b]	1.62	3.30	3.50%	2.33	4.76	5.40%	3.38	6.50	8.30%
	FC-LSTM [Li *et al.*, 2018b]	2.05	4.19	4.80%	2.20	4.55	5.20%	2.37	4.96	5.70%
	WaveNet [Oord *et al.*, 2016]	1.39	3.01	2.91%	1.83	4.21	4.16%	2.35	5.43	5.87%
	DCRNN [Li *et al.*, 2018b]	1.38	2.95	2.90%	1.74	3.97	3.90%	2.07	4.74	4.90%
	GGRU [Zhang *et al.*, 2018]	-	-	-	-	-	-	-	-	-
	STGCN [Yu *et al.*, 2018]	1.36	2.96	2.90%	1.81	4.27	4.17%	2.49	5.69	5.79%
	Graph WaveNet	**1.30**	**2.74**	**2.73**%	**1.63**	**3.70**	**3.67**%	**1.95**	**4.52**	**4.63**%

Table 2: Performance comparison of Graph WaveNet and other baseline models. Graph WaveNet achieves the best results on both datasets.

- FC-LSTM Recurrent neural network with fully connected LSTM hidden units [Li *et al.*, 2018b].

- WaveNet. A convolution network architecture for sequence data [Oord *et al.*, 2016].

- DCRNN. Diffusion convolution recurrent neural network [Li *et al.*, 2018b], which combines graph convolution networks with recurrent neural networks in an encoder-decoder manner.

- GGRU. Graph gated recurrent unit network [Zhang *et al.*, 2018]. Recurrent-based approaches. GGRU uses attention mechanisms in graph convolution.

- STGCN. Spatial-temporal graph convolution network [Yu *et al.*, 2018], which combines graph convolution with 1D convolution.

4.2 Experimental Setups

Our experiments are conducted under a computer environment with one Intel(R) Core(TM) i9-7900X CPU @ 3.30GHz and one NVIDIA Titan Xp GPU card. To cover the input sequence length, we use eight layers of Graph WaveNet with a sequence of dilation factors 1, 2, 1, 2, 1, 2, 1, 2. We use Equation 4 as our graph convolution layer with a diffusion step $K = 2$. We randomly initialize node embeddings by a uniform distribution with a size of 10. We train our model using Adam optimizer with an initial learning rate of 0.001. Dropout with p=0.3 is applied to the outputs of the graph convolution layer. The evaluation metrics we choose include mean absolute error (MAE), root mean squared error (RMSE), and mean absolute percentage error (MAPE). Missing values are excluded both from training and testing.

4.3 Experimental Results

Table 2 compares the performance of Graph WaveNet and baseline models for 15 minutes, 30 minutes and 60 minutes ahead prediction on METR-LA and PEMS-BAY datasets. Graph WaveNet obtains the superior results on both datasets.
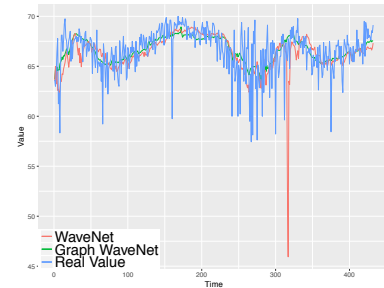

Figure 4: Comparison of prediction curves between WaveNet and Graph WaveNet for 60 minutes ahead prediction on a snapshot of the test data of METR-LA.

It outperforms temporal models including ARIMA, FC-LSTM, and WaveNet by a large margin. Compared to other spatial-temporal models, Graph WaveNet surpasses the previous convolution-based approach STGCN significantly and excels recurrent-based approaches DCRNN and GGRU at the same time. In respect of the second best model GGRU as suggested in Table 2, Graph WaveNet achieves small improvement over GGRU on the 15-minute horizons; however, realizes bigger enhancement on the 60-minute horizons. We think this is because our architecture is more capable of detecting spatial dependencies at each temporal stage. GGRU uses recurrent architectures in which parameters of the GCN layer are shared across all recurrent units. In contrast, Graph WaveNet employs stacked spatial-temporal layers which contain separate GCN layers with different parameters. Therefore each GCN layer in Graph WaveNet is able to focus on its own range of temporal inputs.

We plot 60-minutes-ahead predicted values v.s real values of Graph WaveNet and WaveNet on a snapshot of the test data in Figure 4. It shows that Graph WaveNet generates more stable predictions than WaveNet. In particular, there is a red sharp spike produced by WaveNet, which deviates far from real values. On the contrary, the curve of Graph WaveNet goes in the middle of real values all the time.

Dataset	Model Name	Adjacency Matrix Configuration	Mean MAE	Mean RMSE	Mean MAPE
METR-LR	Identity	$[\mathbf{I}]$	3.58	7.18	10.21%
	Forward-only	$[\mathbf{P}]$	3.13	6.26	8.65%
	Adaptive-only	$[\tilde{\mathbf{A}}_{adp}]$	3.10	6.21	8.68%
	Forward-backward	$[\mathbf{P}_f, \mathbf{P}_b]$	3.08	6.13	8.25%
	Forward-backward-adaptive	$[\mathbf{P}_f, \mathbf{P}_b, \tilde{\mathbf{A}}_{adp}]$	**3.04**	**6.09**	**8.23%**
PEMS-BAY	Identity	$[\mathbf{I}]$	1.80	4.05	4.18%
	Forward-only	$[\mathbf{P}_f]$	1.62	3.61	3.72%
	Adaptive-only	$[\tilde{\mathbf{A}}_{adp}]$	1.61	3.63	3.59%
	Forward-backward	$[\mathbf{P}_f, \mathbf{P}_b]$	1.59	3.55	3.57%
	Forward-backward-adaptive	$[\mathbf{P}_f, \mathbf{P}_b, \tilde{\mathbf{A}}_{adp}]$	**1.58**	**3.52**	**3.55%**

Table 3: Experimental results of different adjacency matrix configurations. The forward-backward-adaptive model achieves the best results on both datasets. The adaptive-only model achieves nearly the same performance with the forward-only model.
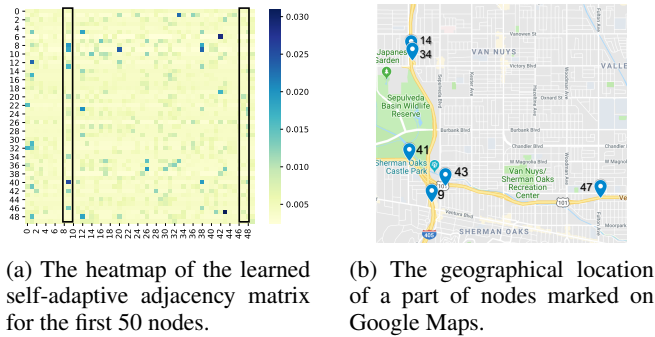

(a) The heatmap of the learned self-adaptive adjacency matrix for the first 50 nodes.

(b) The geographical location of a part of nodes marked on Google Maps.

Figure 5: The learned self-adaptive adjacency matrix.

Effect of the Self-Adaptive Adjacency Matrix

To verify the effectiveness of our proposed adaptive adjacency matrix, we conduct experiments with Graph WaveNet using five different adjacency matrix configurations. Table 3 shows the average score of MAE, RMSE, and MAPE over 12 prediction horizons. We find that the adaptive-only model works even better than the forward-only model with mean MAE. When the graph structure is unavailable, Graph WaveNet would still be able to realize a good performance. The forward-backward-adaptive model achieves the lowest scores on all three evaluation metrics. It indicates that if graph structural information is given, adding the self-adaptive adjacency matrix could introduce new and useful information to the model. In Figure 5, we further investigate the learned self-adaptive adjacency matrix under the configuration of the forward-backward-adaptive model trained on the METR-LA dataset. According to Figure 5a, some columns have more high-value points than others such as column 9 in the left box compared to column 47 in the right box. It suggests that some nodes are influential to most nodes in a graph while other nodes have weaker impacts. Figure 5b confirms our observation. It can be seen that node 9 locates nearby the intersection of several main roads while node 47 lies in a single road.

Computation Time

We compare the computation cost of Graph WaveNet with DCRNN and STGCN on the METR-LA dataset in Table 4.

Model	Computation Time	
	Training(s/epoch)	Inference(s)
DCRNN	249.31	18.73
STGCN	19.10	11.37
Graph WaveNet	53.68	2.27

Table 4: The computation cost on the METR-LA dataset.

Graph WaveNet runs five times faster than DCRNN but two times slower than STGCN in training. For inference, we measure the total time cost of each model on the validation data. Graph WaveNet is the most efficient of all at the inference stage. This is because that Graph WaveNet generates 12 predictions in one run while DCRNN and STGCN have to produce the results conditioned on previous predictions.

5 Conclusion

In this paper, we present a novel model for spatial-temporal graph modeling. Our model captures spatial-temporal dependencies efficiently and effectively by combining graph convolution with dilated casual convolution. We propose an effective method to learn hidden spatial dependencies automatically from the data. This opens up a new direction in spatial-temporal graph modeling where the dependency structure of a system is unknown but needs to be discovered. On two public traffic network datasets, Graph WaveNet achieves state-of-the-art results. In future work, we will study scalable methods to apply Graph WaveNet on large-scale datasets and explore approaches to learn dynamic spatial dependencies.

Acknowledgments

References

[Atwood and Towsley, 2016] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *NIPS*, pages 1993–2001, 2016.

[Bruna *et al.*, 2014] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

[Dauphin *et al.*, 2017] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *ICML*, pages 933–941, 2017.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016.

[Gilmer *et al.*, 2017] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, pages 1263–1272, 2017.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1024–1034, 2017.

[Jain *et al.*, 2016] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *CVPR*, pages 5308–5317, 2016.

[Kalchbrenner *et al.*, 2016] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Li *et al.*, 2018a] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *AAAI*, pages 3546–3553, 2018.

[Li *et al.*, 2018b] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*, 2018.

[Liu *et al.*, 2019] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *AAAI*, 2019.

[Monti *et al.*, 2017] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pages 5115–5124, 2017.

[Oord *et al.*, 2016] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[Pan *et al.*, 2018] Shirui Pan, Ruiqi Hu, Sai-fu Fung, Guodong Long, Jing Jiang, and Chengqi Zhang. Learning graph embedding with adversarial training methods. In *IJCAI*, 2018.

[Seo *et al.*, 2018] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *NIPS*, pages 362–373, 2018.

[Shuman *et al.*, 2012] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *arXiv preprint arXiv:1211.0053*, 2012.

[Velickovic *et al.*, 2017] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2017.

[Wang *et al.*, 2017] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *CIKM*, pages 889–898. ACM, 2017.

[Wu *et al.*, 2019] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[Yan *et al.*, 2018] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, pages 3482–3489, 2018.

[Yao *et al.*, 2018] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. Deep multi-view spatial-temporal network for taxi demand prediction. In *AAAI*, pages 2588–2595, 2018.

[Ying *et al.*, 2018] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NIPS*, pages 4800–4810, 2018.

[Yu and Koltun, 2016] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[Yu *et al.*, 2018] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*, pages 3634–3640, 2018.

[Zhang and Chen, 2018] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *NIPS*, pages 5165–5175, 2018.

[Zhang *et al.*, 2018] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*, 2018.