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

项雅丽

2019年9月11日

Motivation

- Existing approaches mostly capture the spatial dependency on a **fixed graph structure**,
- explicit graph structure (relation) **does not** necessarily reflect the true dependency
- genuine relation may be **missing** due to the incomplete connections in the data
- RNN become inefficient for long sequences and its gradients are more likely to explode when they are combined with graph convolution networks.

- node's future information is conditioned on its historical information as well as its neighbors' historical information.
- each node has **dynamic input features**. The aim is to **model each node's dynamic features** given the graph structure.

Problem Definition

dynamic feature matrix:

$$\mathbf{X}^{(t)} \in \mathbf{R}^{N \times \bar{D}}.$$

task:

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

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

Graph Convolution Layer

GCN:

$$\mathbf{Z} = \tilde{\mathbf{A}}\mathbf{X}\mathbf{W}.$$

Diffusion convolution layer, effective in spatial-temporal modeling:

$$\mathbf{Z} = \sum_{k=0}^{K} \mathbf{P}^{k}\mathbf{X}\mathbf{W}_{\mathbf{k}},$$

For directed graph:

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

$$\mathbf{P} = \mathbf{A}/rowsum(\mathbf{A})$$

$$\mathbf{P}_f = \mathbf{A}/rowsum(\mathbf{A})$$

$$\mathbf{P}_b = \mathbf{A}^{\mathbf{T}}/rowsum(\mathbf{A}^{\mathbf{T}})$$

self-adaptive adjacency matrix

This self-adaptive adjacency matrix does not require any prior knowledge and is learned end-to-end through stochastic gradient descent:

$$\mathbf{E}_1, \mathbf{E}_2 \in \mathbf{R}^{N \times c}.$$

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

can be considered as the transition matrix of a hidden diffusion process

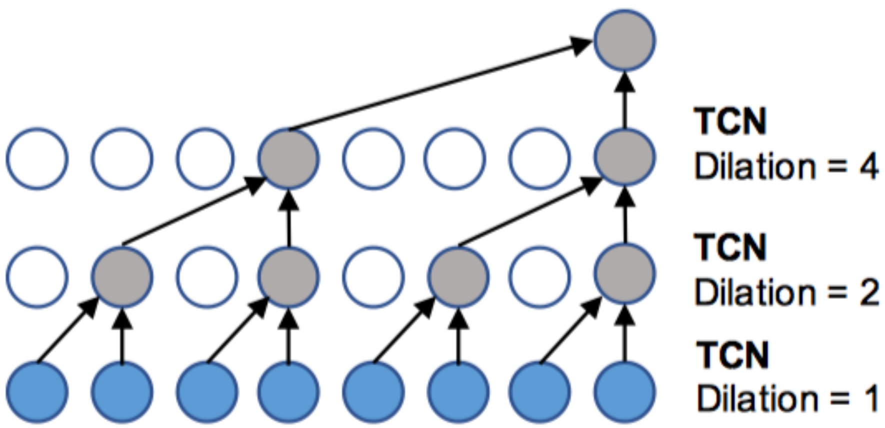
$$\mathbf{Z} = \sum_{k=0}^{K} \mathbf{P}_f^k \mathbf{X} \mathbf{W}_{k1} + \mathbf{P}_b^k \mathbf{X} \mathbf{W}_{k2} + \tilde{\mathbf{A}}_{apt}^k \mathbf{X} \mathbf{W}_{k3}.$$

When the graph structure is unavailable, use the **self-adaptive adjacency matrix alone** to capture hidden spatial dependencies:

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

Temporal Convolution Layer

dilated causal convolution



- parallel computation
- alleviates the gradient explosion

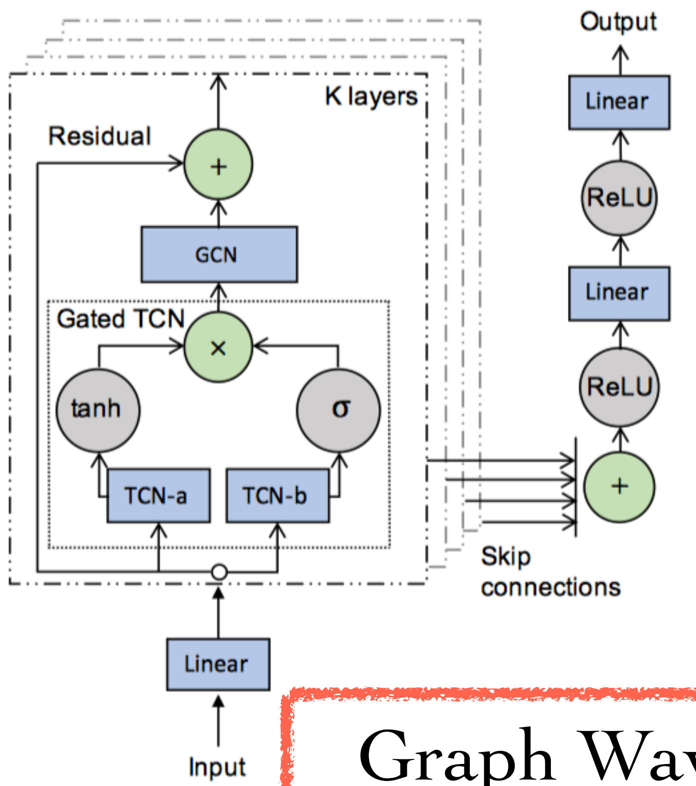
Figure 2: Dilated casual convolution with kernel size 2. With a dilation factor k, it picks inputs every k step and applies the standard 1D convolution to the selected inputs.

$$\mathbf{x} \in \mathbf{R}^{T}$$

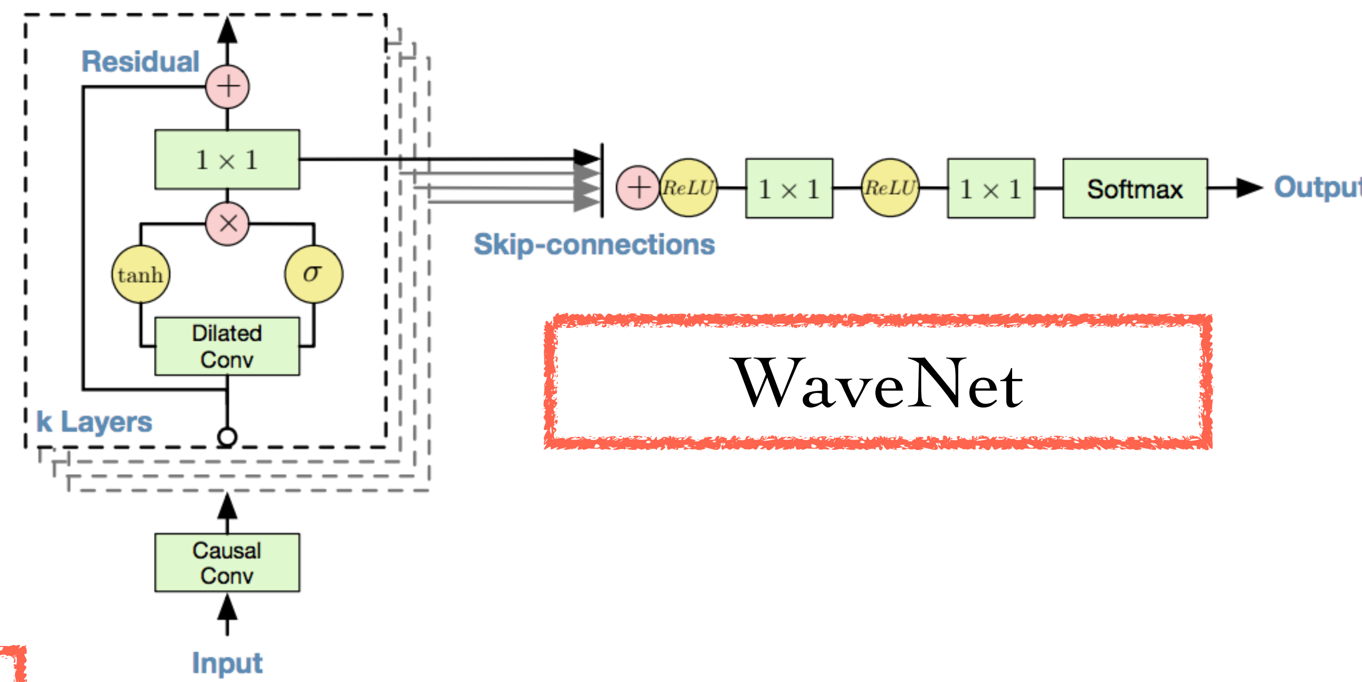
$$\text{filter } \mathbf{f} \in \mathbf{R}^{K}$$

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

Framework of Graph WaveNet



Graph WaveNet



WaveNet

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.

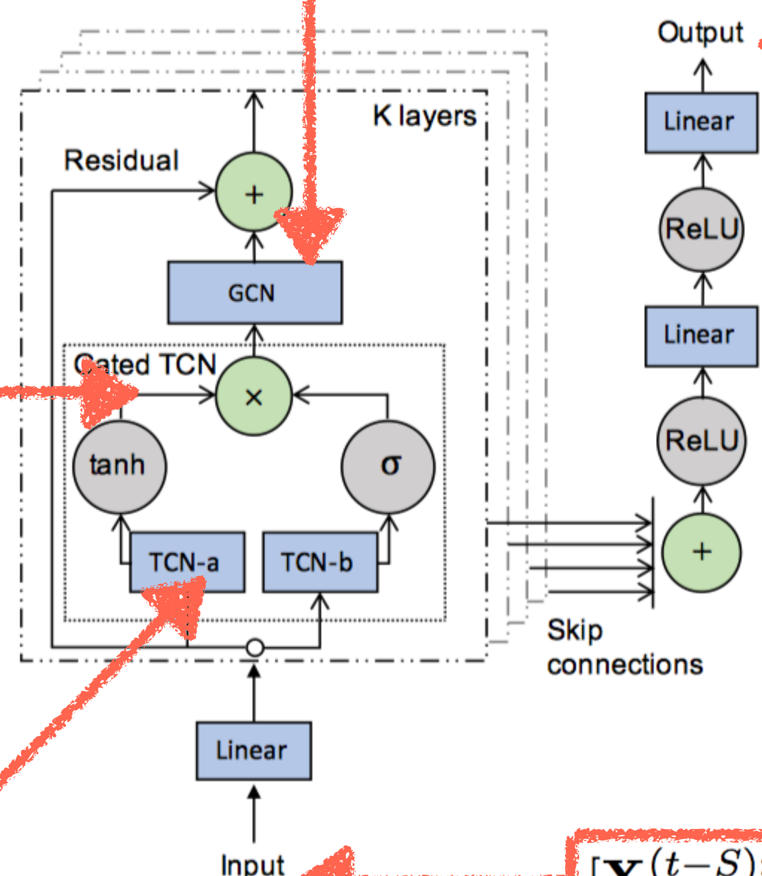
$$L(\hat{\mathbf{X}}^{(t+1):(t+T)};\mathbf{\Theta}) = \frac{1}{TND}\sum_{i=1}^{i=T}\sum_{j=1}^{j=N}\sum_{k=1}^{k=D}|\hat{\mathbf{X}}_{jk}^{(t+i)} - \mathbf{X}_{jk}^{(t+i)}|$$

Graph WaveNet outputs $X^{(t+1):(t+T)}$ as a whole rather than generating $X^{(t)}$ recursively through T steps.

Framework of Graph WaveNet

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

$$\mathbf{X}^{(t+1):(t+T)} \in \mathbf{R}^{N \times D \times T}$$



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

$$[\mathbf{X}^{(t-S):t}, G]$$

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

Experiments

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.

Experiments

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.

谢谢 :)