
O
G
D
O
O
O
O
D

Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting

2. Preliminary

2.1 Traffic Prediction on Road Graphs

→ predict the most likely traffic measurements (e.g. speed or traffic flow) in the next H steps given the previous M traffic observation,

$$\hat{v}_{t+1}, \dots, \hat{v}_{t+H} = \underset{v_{t+1}, \dots, v_{t+H}}{\arg\max} \log P(v_{t+1}, \dots, v_{t+H} \mid v_{t-M+1}, \dots v_t), \quad (1)$$

→ $v_t \in \mathbb{R}^n$ is an observation vector of n road segments at time step t

→ $G_t = (V_t, \mathcal{E}, W)$, V_t is a finite set of vertices, from n monitor stations in traffic network, $\mathcal{E}$, is a set of edges, indicating the connectedness between stations, while $W \in \mathbb{R}^{n \times n}$ denotes the weighted adjacency matrix of G_t.

2.2 Convolution on graph

→ graph convolution operator "$*_G$" based on spectral graph convolution

Convolutional Theorem in graph fourier transform

- make graph undirected

- $L = D - A$ [Laplacian matrix]
  ↳ intuition = difference between graph signal X_i and its $\text{avg}(\{X_j \mid j \in N_i\})$

→ normalized symmetric Laplacian $(\tilde{L}) = D^{-\frac{1}{2}} . L . D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} . A . D^{-\frac{1}{2}}$

→ because L is symmetric, eigenvectors are orthonormal, i.e. $U_i^T U_j = 0$ if $i \neq j$, and $U_i^T U_i = 1$. In matrix form, this becomes $UU^T = U^T U = I$ hence:

$$L = U . \Lambda . U^T$$

eigenvalues along the diagonal matrix of L

eigenvectors of L

→ fourier transform of signal $X \in \mathbb{R}^n$: (graph G with laplacian L)
$\hat{f} = \mathcal{F}\{X\} = U^T . X$

eigenvectors of L

→ inverse fourier transform: $\mathcal{F}^{-1}\{x\} = U.\hat{f} = U.\mathcal{F}\{X\}$

→ "$*_G$" convolution operator specific to graph G:
$\mathcal{F}\{X *_G W\} = \mathcal{F}\{f\} \odot \mathcal{F}\{g\} \Leftrightarrow X *_G W = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \odot \mathcal{F}\{g\}\}$
$X *_G W = U(U^T.X \odot U^T.W) = U(U^T.X \odot \hat{W}_\theta)$

hadamard/element-wise product

$= (U.\hat{W}_\theta.U^T).X$

spectral filter
$\hat{W}_\theta = U^T.W \in \mathbb{R}^n$
W = weight matrix
X = graph signal
$\hat{W}_\theta = \mathrm{diag}(\theta_1, \dots, \theta_n)$
adaptive parameters in fourier domain

→ Approximate spectral filter with $p_N(\Lambda)$, a degree N polynomial of the eigenvalues of the laplacian L. Why? Convolution in time domain = element-wise multiplication in fourier/frequency domain, so to ensure $\hat{W}$ to be meaningful, parameterize $\hat{W}_\theta$ based on eigenvalues of the laplacian L.

→ analogy:

	fourier	graph
1.	modes/basis function	eigenvectors of L
2	frequencies	eigenvalues of L

$$X *_G W = (U.p_N(\Lambda).U^T).X = p_N(U.\Lambda.U^T).X \quad \text{(eq. 1)}$$
$$= p_N(L).X^T \quad (3)$$

→ evaluating equation 1 is expensive $O(N^2)$

→ We can approximate $p_N(\Lambda)$ with:

$$p_N(\Lambda) \approx \sum_{k=0}^{K-1} \theta'_k . T_k(\tilde{\Lambda}) \quad \text{, where } \tilde{\Lambda} = \frac{2}{\lambda_{max}} . \Lambda - I_N$$

largest eigenvalue

$$T_k(x) = 2x.T_{k-1}(x) - T_{k-2}(x)$$
with $T_0(x) = 1$. & $T_1(x) = x$

→ hence with Chebyshev Polynomial Approximations, we can rewrite $X *_G . W$ as

$$X *_G W \approx \sum_{k=0}^{K-1} \theta'_k . T(\tilde{L}).X \quad \text{(eq. 5)}$$

→ where $T_k(\tilde{L}) \in \mathbb{R}^{N \times N}$ is the Chebysev polynomial approximation of order k evaluated at laplacian $\tilde{L} = \frac{2L}{\lambda_{max}} - I_N$

→ now convolution time complexity is $O(K|\mathcal{E}|)$

→ eq.(5) is now K-localized since it is K^{th} order polynomial in the Laplacian, i.e. it depends only on nodes that are maximum K-hop away from the central node.

1-st order approximation

→ assume $\lambda_{max} = 2$

$$X *_g W \simeq \theta_0' X + \theta_1' \left(\frac{2}{\lambda_{max}} . L - I_N\right) X$$

$$\simeq \theta_0' X - \theta_1' (D^{-\frac{1}{2}} . W . D^{-\frac{1}{2}}) . X$$

→ replace $\theta = \theta_0 = -\theta_1$, $\tilde{W} = W + I_N$, and $\tilde{D}_{ii} = \sum_j \tilde{W}_{ij}$

$$X *_g W = \theta (I_N + D^{-\frac{1}{2}} . W D^{-\frac{1}{2}}) . X = \theta (\tilde{D}^{-\frac{1}{2}} . \tilde{W} . \tilde{D}^{-\frac{1}{2}}) . X$$

→ in this problem :- $W \in \mathbb{R}^{K \times C_i \times C_o}$ } C_i, C_o = size of input & output feature maps

- for each time step t of M, convolution operation with same kernel W imposed on $X_t \in \mathbb{R}^{n \times C_i}$ in parallel, thus X in convolution is $X \in \mathbb{R}^{M \times n \times C_i}$
- frame $v_t \in \mathbb{R}^n = X \in \mathbb{R}^{n \times C_i}$ $[C_i = 1]$

3. Proposed models

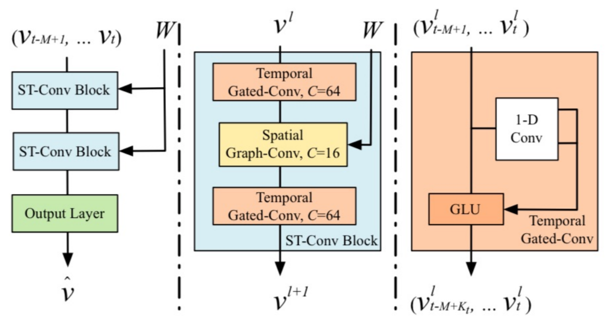

Figure 2: Architecture of spatio-temporal graph convolutional networks. The framework STGCN consists of two spatio-temporal convolutional blocks (ST-Conv blocks) and a fully-connected output layer in the end. Each ST-Conv block contains two temporal gated convolution layers and one spatial graph convolution layer in the middle. The residual connection and bottleneck strategy are applied inside each block. The input $v_{t-M+1}, ..., v_t$ is uniformly processed by ST-Conv blocks to explore spatial and temporal dependencies coherently. Comprehensive features are integrated by an output layer to generate the final prediction $\hat{v}$.

3.1. Gated-CNN for Extracting Temporal Features

→ RNN for traffic prediction = time consuming iterations
→ Gated-CNN = faster than RNN (allow parallelization)
→ capture & extract temporal dynamic behaviors of traffic flows.

→ kernel $\Gamma \in \mathbb{R}^{K_t \times C_i \times 2C_o}$ } $[P\,Q] = \Gamma * Y \in \mathbb{R}^{(M-K_t+1) \times 2C_o}$
1D-Convolution
→ input $Y \in \mathbb{R}^{M \times C_i}$
→ $\Gamma *_T Y = P \odot \sigma(Q) \in \mathbb{R}^{(M-K_t+1) \times C_o}$
Sigmoid

→ intuition: 1. extracting temporal traffic flow features.

2. $\sigma(\cdot) \in (0,1)$, control which input P of current states are relevant for discovering compositional structure and dynamic variance in time-series.

3. $\nabla(\Gamma *_T Y) = \nabla P \odot \sigma(Q) + \nabla\sigma(Q) \odot P$. when $\sigma(\cdot) \approx 0$, second term not eq. to 0 & when $\sigma(\cdot) \approx 1$, first term not eq. to 0, thus solve vanishing gradient problem

3.2. Spatio-Temporal Convolutional Block

→ fuse features from both spatial and temporal domains

→ input $v^l \in \mathbb{R}^{M \times n \times C^l}$ of block l

→ output $v^{l+1} \in \mathbb{R}^{(M-2(K_t-1)) \times n \times C^{l+1}}$

$$v^{l+1} = \Gamma_1^l *_T \mathrm{RELU}(W_\theta^l *_G (\Gamma_0^l *_T v^l))$$

→ add 2 fully-connected layer / output layer after 2 ST-Conv block

→ Loss function:

$$L(\hat{v}, W_\theta') = \sum_t \| \hat{v}(v_{t-M+1}, \ldots, v_t, W_\theta') - v_{t+1} \|^2$$

model prediction

3.3. Data Preprocessing

→ weighted adjacency matrix W:

$$w_{ij} = \begin{cases} \exp\left(-\frac{d_{ij}^2}{\sigma^2}\right), & i \neq j \text{ and } \exp\left(-\frac{d_{ij}^2}{\sigma^2}\right) \geq \varepsilon \\ 0, & \text{otherwise} \end{cases}$$

→ d_{ij} = distance between station i and j.

4. Experiments

→ PEMS = 39k sensor stations, deployed across California. 5-minute interval road segment speed.

→ PEMS7(L) = 1026 stations in district 7.

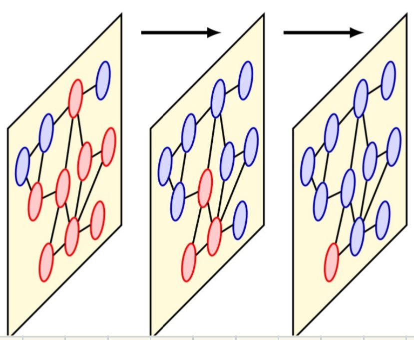

Figure 13.4 Schematic illustration of information flow through successive layers of a graph neural network. In the third layer a single node is highlighted in red. It receives information from its two neighbours in the previous layer and those in turn receive information from their neighbours in the first layer. As with convolutional neural networks for images, we see that the effective receptive field, corresponding to the number of nodes shown in red, grows with the number of processing layers.

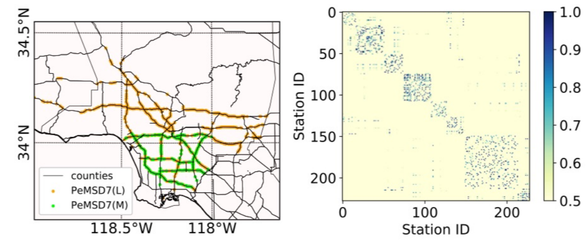

Figure 3: PeMS sensor network in District 7 of California (left), each dot denotes a sensor station; Heat map of weighted adjacency matrix in PeMSD7(M) (right).

Model	BJER4 (15/ 30/ 45 min)		
	MAE	MAPE (%)	RMSE
HA	5.21	14.64	7.56
LSVR	4.24/ 5.23/ 6.12	10.11/ 12.70/ 14.95	5.91/ 7.27/ 8.81
ARIMA	5.99/ 6.27/ 6.70	15.42/ 16.36/ 17.67	8.19/ 8.38/ 8.72
FNN	4.30/ 5.33/ 6.14	10.68/ 13.48/ 15.82	5.86/ 7.31/ 8.58
FC-LSTM	4.24/ 4.74/ 5.22	10.78/ 12.17/ 13.60	5.71/ 6.62/ 7.44
GCGRU	3.84/ 4.62/ 5.32	9.31/ 11.41/ 13.30	5.22/ 6.35/ 7.58
STGCN(Cheb)	**3.78/ 4.45/ 5.03**	**9.11/ 10.80/ 12.27**	**5.20/ 6.20/ 7.21**
STGCN(1^{st})	3.83/ 4.51/ 5.10	9.28/ 11.19/ 12.79	5.29/ 6.39/ 7.39

Table 1: Performance comparison of different approaches on the dataset BJER4.

Model	PeMSD7(M) (15/ 30/ 45 min)			PeMSD7(L) (15/ 30/ 45 min)		
	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE
HA	4.01	10.61	7.20	4.60	12.50	8.05
LSVR	2.50/ 3.63/ 4.54	5.81/ 8.88/ 11.50	4.55/ 6.67/ 8.28	2.69/ 3.85/ 4.79	6.27/ 9.48/ 12.42	4.88/ 7.10/ 8.72
ARIMA	5.55/ 5.86/ 6.27	12.92/ 13.94/ 15.20	9.00/ 9.13/ 9.38	5.50/ 5.87/ 6.30	12.30/ 13.54/ 14.85	8.63/ 8.96/ 9.39
FNN	2.74/ 4.02/ 5.04	6.38/ 9.72/ 12.38	4.75/ 6.98/ 8.58	2.74/ 3.92/ 4.78	7.11/ 10.89/ 13.56	4.87/ 7.02/ 8.46
FC-LSTM	3.57/ 3.94/ 4.16	8.60/ 9.55/ 10.10	6.20/ 7.03/ 7.51	4.38/ 4.51/ 4.66	11.10/ 11.41/ 11.69	7.68/ 7.94/ 8.20
GCGRU	2.37/ 3.31/ 4.01	5.54/ 8.06/ 9.99	4.21/ 5.96/ 7.13	2.48/ 3.43/ 4.12 *	5.76/ 8.45/ 10.51 *	4.40/ 6.25/ 7.49 *
STGCN(Cheb)	**2.25/ 3.03/ 3.57**	5.26/ **7.33/ 8.69**	**4.04/ 5.70/ 6.77**	**2.37/ 3.27/ 3.97**	**5.56/ 7.98/ 9.73**	**4.32/ 6.21/ 7.45**
STGCN(1^{st})	2.26/ 3.09/ 3.79	**5.24**/ 7.39/ 9.12	4.07/ 5.77/ 7.03	2.40/ 3.31/ 4.01	5.63/ 8.21/ 10.12	4.38/ 6.43/ 7.81

Table 2: Performance comparison of different approaches on the dataset PeMSD7.

Dataset	Time Consumption (s)		
	STGCN(Cheb)	STGCN(1^{st})	GCGRU
PeMSD7(M)	**272.34**	271.18	3824.54
PeMSD7(L)	1926.81	**1554.37**	19511.92

Table 3: Time consumptions of training on the dataset PeMSD7.

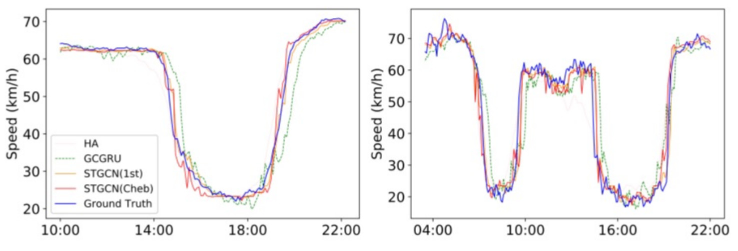

Figure 4: Speed prediction in the morning peak and evening rush hours of the dataset PeMSD7.

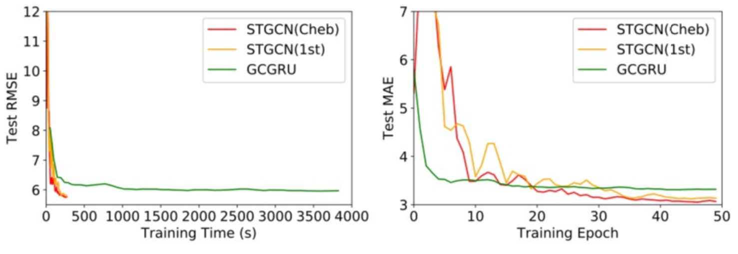

Figure 5: Test RMSE versus the training time (left); Test MAE versus the number of training epochs (right). (PeMSD7(M))

References:

1. STGCN, Yu et al.
2. GCN, Kipf et al.
3. Graph Representation Learning, Hamilton et al.