Abstract
Spatiotemporal forecasting has various applications in neuroscience, climate and transportation domain. Traffic forecasting is one canonical example of such learning task. The task is challenging due to (1) complex spatial dependency on the road network, (2) non-linear temporal dynamics with changing road conditions and (3) inherent difficulty of long-term forecasting. To address these challenges, we propose to model the traffic flow as a diffusion process on a directed graph and introduce Diffusion Convolutional Recurrent Neural Network (DCRNN), a deep learning framework for traffic forecasting that incorporates both spatial and temporal dependency in the traffic flow. Specifically, DCRNN captures the spatial dependency using bidirectional random walks on the graph, and the temporal dependency using the encoder-decoder architecture with scheduled sampling. We evaluate the framework on two real-world large scale road network traffic datasets and observe consistent improvement of 12% - 15% over state-of-the-art baselines.

1 Introduction
Spatiotemporal forecasting is a crucial task for a learning system that operates in a dynamic environment. It has a wide range of applications from autonomous vehicles operations, to energy and smart grid optimization, and to logistics and supply chain management. In this paper, we study one important task: traffic forecasting on road networks, the core component of the intelligent transportation systems. The goal of traffic forecasting is to predict the future traffic speeds of a sensor network given historic traffic speeds and the underlying road network.

This task is challenging mainly due to the complex spatiotemporal dependencies and inherent difficulty in the long term forecasting. On the one hand, traffic time series demonstrate strong temporal dynamics. Recurring incidents such as rush hours or accidents can cause non-stationarity, making it difficult to forecast long-term. On the other hand, sensors on the road network contain complex yet unique spatial correlations. Figure 1 illustrates an example. Road segments 1 and 2 are correlated, road 3 is not. Although 1 and 3 are close in the Euclidean space, they demonstrate very different behaviors. Moreover, the future traffic speed is influenced more by the downstream traffic than the upstream one. This shows that the spatial structure in traffic is non-Euclidean and directional.

Figure 1: Spatial correlation is dominated by road network structure. (1) Traffic speed in road 1 and road 2 are similar as they locate in the same highway. (2) Road 1 and road 3 differ significantly, as they locate in the opposite directions. Though close to each other in the Euclidean space, their road network distance is large.
Traffic forecasting has been studied for decades, falling into two main categories: knowledge-driven approach and data-driven approach. In transportation and operational research, knowledge-driven methods usually apply queuing theory and simulate user behaviors in traffic [4]. In time series community, data-driven methods such as autoregressive integrated moving average (ARIMA) model and Kalman filtering remain popular [14, 13]. However, simple time series models usually rely on the stationarity assumption, which is often violated by the traffic data. Most recently, deep learning models for traffic forecasting have been developed in [15, 25], but without considering the spatial structure. In [22] and [16], the authors model the spatial correlation with Convolutional Neural Networks (CNN), but the spatial structure is in the Euclidean space (e.g., 2D images). In [3] and [6], the authors studied graph convolution, but only for undirected graphs.

In this work, we represent the pair-wise spatial correlations between traffic sensors as a directed graph whose nodes are sensors and edge weights denote proximity between the sensor pairs measured by road network distance. We model the dynamics of the traffic flow as a diffusion process and propose the diffusion convolution operation to capture the spatial dependency. Our Diffusion Convolutional Recurrent Neural Network (DCRNN) integrates diffusion convolution, the sequence to sequence architecture and the scheduled sampling technique. When evaluated on real-world traffic datasets, DCRNN consistently outperforms state-of-the-art traffic forecasting baselines by a large margin.

2 Methodology

We formalize the learning problem of spatiotemporal traffic forecasting and describe how to model the dependency structures using diffusion convolutional recurrent neural network.

Traffic Forecasting Problem The goal of traffic forecasting is to predict the future traffic speed given previously observed traffic flow from $N$ correlated sensors on the road network. We can represent the sensor network as a weighted directed graph $G = (V, E, W)$. Here $V$ is a set of nodes, i.e., sensors, with $|V| = N$, $E$ is a set of edges and $W \in \mathbb{R}^{N \times N}$ is a weighted adjacency matrix representing the nodes proximity (e.g., a function of their road network distance). Denote the traffic flow observed on $G$ as a graph signal $X \in \mathbb{R}^{N \times P}$, where $P$ is the number of features of each node (e.g., velocity, volume). Let $X^{(t)}$ represent the graph signal observed at time $t$, the traffic forecasting problem aims to learn a function $h(\cdot)$ that maps $T'$ historical graph signals to future $T$ graph signals, given a graph $G$:

$$[X^{(t-T'+1)}, \ldots, X^{(t)}; G] \xrightarrow{h(\cdot)} [X^{(t+1)}, \ldots, X^{(t+T)}]$$

2.1 Spatial Dependency Modeling

We model the spatial dependency by relating traffic flow to a diffusion process, which explicitly captures the stochastic nature of traffic dynamics. This diffusion process is characterized by a random walk on $G$ with restart probability $\alpha \in [0, 1]$ and a transition matrix $D^{-1}_G W$. Here $D_G = \text{diag}(W1)$
is the out-degree diagonal matrix, and $1 \in \mathbb{R}^N$ denotes the all one vector. The stationary distribution of the diffusion process can be represented as a weighted combination of infinite random walks on the graph [20]. In practice, we use a finite $K$-step truncation of the diffusion process and assign a trainable weight to each step. We also include the reverse direction diffusion process, such that the bidirectional diffusion offers the model more flexibility to capture the influence from both the upstream and the downstream traffic. The resulted diffusion convolution operation over a graph signal $X \in \mathbb{R}^{N \times T}$ and a filter $f_\theta$ is defined as:

$$X_{:,p} \ast_{\mathcal{G}} f_\theta = \sum_{k=0}^{K-1} \left( \theta_{k,1} \left( D^{-1}_O W \right)^k + \theta_{k,2} \left( D^{-1}_I W^T \right)^k \right) X_{:,p} \quad \text{for} \quad p \in \{1, \cdots, P\} \quad (1)$$

where $\theta \in \mathbb{R}^{K \times 2}$ are the parameters for the filter and $D^{-1}_O W, D^{-1}_I W^T$ represent the transition matrices of the diffusion process and the reverse one respectively. In general, computing the convolution can be expensive. However, if $\mathcal{G}$ is sparse, Equation (1) can be calculated efficiently using $O(K|\mathcal{E}|) \ll O(N^2)$. See Appendix [B] for more detail.

Diffusion convolution is defined on both directed and undirected graphs. When applied to undirected graphs, we show that many existing graph structured convolutional operations including the popular spectral graph convolution, i.e., ChebNet [6], can be considered as a special case of diffusion convolution (up to a similarity transformation). Let $D$ denote the degree matrix, and $L = D^{-\frac{1}{2}} (D - W) D^{-\frac{1}{2}}$ be the normalized graph Laplacian, the following Proposition demonstrates the connection.

**Proposition 2.1.** The spectral graph convolution defined as

$$X_{:,p} \ast_{\mathcal{G}} f_\theta = \Phi(\theta) \Phi^T X_{:,p}$$

with eigenvalue decomposition $L = \Phi \Lambda \Phi^T$ and $F(\theta) = \sum_{k=0}^{K-1} \theta_k \Lambda^k$, is equivalent to diffusion convolution up to a similarity transformation, when the graph $\mathcal{G}$ is undirected.

**Proof.** See Appendix [C]

### 2.2 Temporal Dynamics Modeling

We leverage the recurrent neural networks (RNNs) to model the temporal dependency. In particular, we use Gated Recurrent Units (GRU) [5], which is a simple yet powerful variant of RNNs. We replace the matrix multiplications in GRU with the diffusion convolution, which leads to our proposed *Diffusion Convolutional Gated Recurrent Unit (DCGRU)*.

$$r^{(t)} = \sigma(\Theta_r \ast_{\mathcal{G}} [X^{(t)}, H^{(t-1)}] + b_r) \quad u^{(t)} = \sigma(\Theta_u \ast_{\mathcal{G}} [X^{(t)}, H^{(t-1)}] + b_u) \quad C^{(t)} = \tanh(\Theta_c \ast_{\mathcal{G}} [X^{(t)}, (r^{(t)} \odot H^{(t-1)})] + b_c) \quad H^{(t)} = u^{(t)} \odot H^{(t-1)} + (1 - u^{(t)}) \odot C^{(t)}$$

where $X^{(t)}, H^{(t)}$ denote the input and output of at time $t$, $r^{(t)}, u^{(t)}$ are reset gate and update gate at time $t$ respectively. $\ast_{\mathcal{G}}$ denotes the diffusion convolution defined in Equation (1) and $\Theta_r, \Theta_u, \Theta_c$ are parameters for the corresponding filters.

In multiple-step ahead forecasting, we employ the *Sequence to Sequence* architecture [19] with *Scheduled Sampling* [2]. The diffusion convolution is applied to both the encoder and the decoder. With both spatial and temporal modeling, we build a *Diffusion Convolutional Recurrent Neural Network (DCRNN)* with architecture shown in Figure [2]. The entire network is trained by maximizing the likelihood of generating the target future time series using backpropagation through time.

### 3 Experiments

We conduct experiments on two real-world large-scale datasets: (1) **Metr-LA**, which contains 4 months of traffic information collected from 207 loop detectors in the highway of Los Angeles County in 2012 [11]. (2) **PEMS-BAY**, which contains 6 months of traffic information collected from by 325 highway sensors in the Bay Area by the CalTrans Performance Measurement System (PeMS). We compare DCRNN with traditional time series regression models and deep neural network based approaches. Detailed description of the datasets and the approaches is available in Appendix [E].

Table 1 shows the comparison of different approaches for 15 minutes, 30 minutes and 1 hour ahead forecasting on both datasets. These methods are evaluated based on three commonly used metrics
Table 1: Performance comparison of different approaches for traffic speed forecasting. DCRNN achieves the best performance with all the metrics for all forecasting horizons.

<table>
<thead>
<tr>
<th>Metric</th>
<th>HA</th>
<th>ARIMA</th>
<th>VAR</th>
<th>SVR</th>
<th>FNN</th>
<th>FC-LSTM</th>
<th>DCRNN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>METR-LA</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 min MAE</td>
<td>4.16</td>
<td>3.99</td>
<td>4.22</td>
<td>3.99</td>
<td>3.99</td>
<td>3.44</td>
<td>2.77</td>
</tr>
<tr>
<td>15 min RMSE</td>
<td>7.80</td>
<td>8.21</td>
<td>7.89</td>
<td>8.45</td>
<td>7.94</td>
<td>6.30</td>
<td>5.38</td>
</tr>
<tr>
<td>15 min MAPE</td>
<td>13.0%</td>
<td>9.6%</td>
<td>10.2%</td>
<td>9.3%</td>
<td>9.9%</td>
<td>9.6%</td>
<td>7.3%</td>
</tr>
<tr>
<td>30 min MAE</td>
<td>4.16</td>
<td>5.15</td>
<td>5.41</td>
<td>5.05</td>
<td>4.23</td>
<td>3.77</td>
<td>3.15</td>
</tr>
<tr>
<td>30 min RMSE</td>
<td>7.80</td>
<td>10.45</td>
<td>9.13</td>
<td>10.87</td>
<td>8.17</td>
<td>7.23</td>
<td>6.45</td>
</tr>
<tr>
<td>30 min MAPE</td>
<td>13.0%</td>
<td>12.7%</td>
<td>12.7%</td>
<td>12.1%</td>
<td>12.9%</td>
<td>10.9%</td>
<td>8.8%</td>
</tr>
<tr>
<td>1 hour MAE</td>
<td>4.16</td>
<td>6.90</td>
<td>6.52</td>
<td>6.72</td>
<td>4.49</td>
<td>4.37</td>
<td>3.60</td>
</tr>
<tr>
<td>1 hour RMSE</td>
<td>7.80</td>
<td>13.23</td>
<td>10.11</td>
<td>13.76</td>
<td>8.69</td>
<td>8.69</td>
<td>7.59</td>
</tr>
<tr>
<td>1 hour MAPE</td>
<td>13.0%</td>
<td>17.4%</td>
<td>15.8%</td>
<td>14.0%</td>
<td>13.2%</td>
<td>10.5%</td>
<td></td>
</tr>
<tr>
<td><strong>PEMS-BAY</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 min MAE</td>
<td>2.88</td>
<td>1.62</td>
<td>1.74</td>
<td>1.85</td>
<td>2.20</td>
<td>2.05</td>
<td>1.38</td>
</tr>
<tr>
<td>15 min RMSE</td>
<td>5.59</td>
<td>3.30</td>
<td>3.16</td>
<td>3.59</td>
<td>4.42</td>
<td>4.19</td>
<td>2.95</td>
</tr>
<tr>
<td>15 min MAPE</td>
<td>6.8%</td>
<td>3.5%</td>
<td>3.6%</td>
<td>3.8%</td>
<td>5.19%</td>
<td>4.8%</td>
<td>2.9%</td>
</tr>
<tr>
<td>30 min MAE</td>
<td>2.88</td>
<td>2.33</td>
<td>2.32</td>
<td>2.48</td>
<td>2.30</td>
<td>2.20</td>
<td>1.74</td>
</tr>
<tr>
<td>30 min RMSE</td>
<td>5.59</td>
<td>4.76</td>
<td>4.25</td>
<td>5.18</td>
<td>4.63</td>
<td>4.55</td>
<td>3.97</td>
</tr>
<tr>
<td>30 min MAPE</td>
<td>6.8%</td>
<td>5.4%</td>
<td>4.6%</td>
<td>5.0%</td>
<td>5.5%</td>
<td>5.4%</td>
<td>3.9%</td>
</tr>
<tr>
<td>1 hour MAE</td>
<td>2.88</td>
<td>3.38</td>
<td>2.93</td>
<td>3.28</td>
<td>2.46</td>
<td>2.37</td>
<td>2.07</td>
</tr>
<tr>
<td>1 hour RMSE</td>
<td>5.59</td>
<td>6.50</td>
<td>5.44</td>
<td>7.08</td>
<td>5.98</td>
<td>4.99</td>
<td>4.74</td>
</tr>
<tr>
<td>1 hour MAPE</td>
<td>6.8%</td>
<td>8.3%</td>
<td>6.5%</td>
<td>8.0%</td>
<td>5.89%</td>
<td>5.7%</td>
<td>4.9%</td>
</tr>
</tbody>
</table>

Figure 3: Visualization of learned localized filters centered at different nodes with $K = 3$ on the METR-LA dataset. The star denotes the center, and the colors represent the weights. We can see (1) filters are localized around the center, and (2) the weights diffuse alongside the road network.

Figure 4: Traffic time series forecasting visualization. DCRNN generates smooth prediction and is usually better at predicting the start and end of peak hours.

in traffic forecasting, including (1) Mean Absolute Error (MAE), (2) Mean Absolute Percentage Error (MAPE), and (3) Root Mean Squared Error (RMSE). We observe the following phenomenon in both datasets. (1) RNN-based methods, including FC-LSTM and DCRNN, generally outperform other baselines which emphasize the importance of modeling the temporal dependency. (2) DCRNN achieves the best performance regarding all the metrics for all forecasting horizons, which suggests the effectiveness of spatiotemporal dependency modeling.

Figure 3 visualizes examples of learned filters centered at different nodes. Figure 4 shows the visualization of 1 hour ahead forecasting of different methods. More ablation studies and model visualization are available in Appendix F.

4 Conclusion

In this paper, we formulated the traffic prediction on road network as a spatiotemporal forecasting problem, and proposed the diffusion convolutional recurrent neural network to capture the spatiotemporal dependencies. Specifically, we use bidirectional graph random walk to model spatial dependency and recurrent neural network to capture the temporal dynamics. We further integrated the encoder-decoder architecture and scheduled sampling technique to improve the performance for long-term forecasting. When evaluated on two large-scale real-world traffic datasets, our approach obtained significantly better predictions than baselines.
References


Appendix

A Notation

Table 2: Notation

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathcal{G})</td>
<td>a graph</td>
</tr>
<tr>
<td>(\mathcal{V}, v_i)</td>
<td>nodes of a graph, (</td>
</tr>
<tr>
<td>(\mathcal{E})</td>
<td>edges of a graph</td>
</tr>
<tr>
<td>(W, W_{ij})</td>
<td>weight matrix of a graph and its entries</td>
</tr>
<tr>
<td>(D, D_I, D_O)</td>
<td>undirected degree matrix, In-degree/out-degree matrix</td>
</tr>
<tr>
<td>(L)</td>
<td>normalized graph Laplacian</td>
</tr>
<tr>
<td>(\Phi, \Lambda)</td>
<td>eigen-vector matrix and eigen-value matrix of (L)</td>
</tr>
<tr>
<td>(X, \hat{X} \in \mathbb{R}^{N \times P})</td>
<td>a graph signal, and the predicted graph signal.()</td>
</tr>
<tr>
<td>(X^{(t)} \in \mathbb{R}^{N \times P})</td>
<td>a graph signal at time (t).</td>
</tr>
<tr>
<td>(H \in \mathbb{R}^{N \times Q})</td>
<td>output of the diffusion convolutional layer.</td>
</tr>
<tr>
<td>(f_{\theta}, \theta)</td>
<td>convolutional filter and its parameters.</td>
</tr>
<tr>
<td>(f_{\Theta}, \Theta)</td>
<td>convolutional layer and its parameters.</td>
</tr>
</tbody>
</table>

Table 2 summarizes the main notation used in the paper.

B Efficient Calculation of Equation 1

Equation (1) can be decomposed into two parts with the same time complexity, i.e., one part with \(D_O^{-1}W\) and the other part with \(D_I^{-1}W^T\). Thus we will only show the time complexity of the first part.

Let \(T_k(x) = (D_O^{-1}W)^k x\), The first part of Equation 1 can be rewritten as

\[
\sum_{k=0}^{K-1} \theta_k T_k(X_{:p})
\]

As \(T_{k+1}(x) = D_O^{-1}W T_k(x)\) and \(D_O^{-1}W\) is sparse, it is easy to see that Equation 2 can be calculated using \(O(K)\) recursive sparse-dense matrix multiplication each with time complexity \(O(|E|)\). Consequently, the time complexities of both Equation (1) and Equation (2) are \(O(K|E|)\).

C Relation with Spectral Graph Convolution

Let \(D\) denote the degree matrix, and \(L = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}\) be the normalized graph Laplacian, the following Proposition demonstrates the connection.

**Proposition 4.1.** The spectral graph convolution defined as

\[
X_{:p} *_{\phi} f_{\theta} = \Phi F(\theta) \Phi^T X_{:p}
\]

with eigenvalue decomposition \(L = \Phi \Lambda \Phi^T\) and \(F(\theta) = \sum_{k=0}^{K-1} \theta_k \Lambda^k\), is equivalent to diffusion convolution up to a similarity transformation, when the graph \(G\) is undirected.

**Proof.** The spectral graph convolution utilizes the concept of normalized graph Laplacian \(L = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} = \Phi \Lambda \Phi^T\). ChebNet parametrizes \(f_{\theta}\) to be a \(K\) order polynomial of \(\Lambda\), and calculate it using stable Chebyshev polynomial basis.

\[
X_{:p} *_{\phi} f_{\theta} = \Phi \left( \sum_{k=0}^{K-1} \theta_k \Lambda^k \right) \Phi^T X_{:p} = \sum_{k=0}^{K-1} \theta_k L^k X_{:p} = \sum_{k=0}^{K-1} \hat{\theta}_k T_k(\hat{L}) X_{:p}
\]

where \(T_0(x) = 1, T_1(x) = x, T_k(x) = xT_{k-1}(x) - T_{k-2}(x)\) are the basis of the Chebyshev polynomial. Let \(\lambda_{max}\) denotes the largest eigenvalue of \(L\), \(\hat{L} = \frac{2}{\lambda_{max}} L - I\) represents a rescaling of the
graph Laplacian that maps the eigenvalues from \([0, \lambda_{\text{max}}]\) to \([-1, 1]\) since Chebyshev polynomial forms an orthogonal basis in \([-1, 1]\). Equation 3 can be considered as a polynomial of \(\tilde{L}\) and we will show that the output of ChebNet Convolution is similar to the output of diffusion convolution up to constant scaling factor. Assume \(\lambda_{\text{max}} = 2\) and \(D_I = D_O = D\) for undirected graph.

\[
\tilde{L} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} - I = -D^{-\frac{1}{2}}WD^{-\frac{1}{2}} \sim -D^{-1}W
\]

(4)

\(\tilde{L}\) is similar to the negative random walk matrix, thus the output of Equation 3 is also similar to the output of Equation 1 up to constant scaling factor.

Figure 5: Sensor distribution of the METR-LA and PEMS-BAY dataset.

D Related Work

Traffic forecasting is a classic problem in transportation and operational research which are primarily based on queuing theory and simulations [8]. Data-driven approaches for traffic forecasting have received considerable attention, details can be found in a recent survey paper [21] and the references therein. However, existing machine learning models either impose strong stationary assumptions on the data (e.g., auto-regressive model) or fail to account for highly non-linear temporal dependency (e.g., latent space model [24, 7]). Deep learning models deliver new promise for time series forecasting problem. For example, in [25, 12], the authors study time series forecasting using deep Recurrent Neural Networks (RNN). Convolutional Neural Networks (CNN) have also been applied to traffic forecasting. In [27, 26], the authors convert the road network to a regular 2-D grid and apply traditional CNN to predict crowd flow.

Recently, CNN has been generalized to arbitrary graphs based on the spectral graph theory. Graph convolutional neural networks (GCN) are first introduced in [3], which bridges the spectral graph theory and deep neural networks. In [6], the authors propose ChebNet which improves GCN with fast localized convolutions filters. While in [17], the authors combine ChebNet with Recurrent Neural Networks (RNN) for structured sequence modeling. In [23], the authors model the sensor network as a undirected graph and applied ChebNet and convolutional sequence model [9] to do forecasting. One limitation of the mentioned spectral based convolutions is that they generally require the graph to be undirected to calculate meaningful spectral decomposition. Going from spectral domain to vertex domain, the authors in [11] propose diffusion-convolutional neural network (DCNN) which defines convolution as a diffusion process across each node in a graph-structured input. However, it does not consider the temporal dynamics and mainly deal with static graph settings.

Our approach is different from all those methods due to both the problem settings and the formulation of the graph convolution. We model the sensor network as a weighted directed graph which is more realistic than grid or undirected graph. Besides, the proposed convolution is defined using bidirectional graph random walk and is further integrated with the sequence to sequence learning framework as well as the scheduled sampling to model the long-term temporal dependency.
Detailed Experimental Settings

HA  Historical Average, which models the traffic flow as a seasonal process, and uses weighted average of previous seasons as the prediction. The period used is 1 week, and the prediction is based on aggregated data from previous weeks. For example, the prediction for this Wednesday is the averaged traffic speeds from last four Wednesdays. As the historical average method does not depend on short-term data, its performance is invariant to the small increases in the forecasting horizon.

ARIMA$_{kal}$  : Auto-Regressive Integrated Moving Average model with Kalman filter. The orders are (3, 0, 1), and the model is implemented using the `statsmodel` python package.

VAR  Vector Auto-regressive model [10]. The number of lags is set to 3, and the model is implemented using the `statsmodel` python package.

SVR  Linear Support Vector Regression, the penalty term $C = 0.1$, the number of historical observation is 5.

The following deep neural network based approaches are also included.

FNN  Feed forward neural network with two hidden layers, each layer contains 256 units. The initial learning rate is $1e^{-3}$, and reduces to $1e^{-4}$ every 20 epochs starting at the 50th epoch. In addition, for all hidden layers, dropout with ratio 0.5 and L2 weight decay $1e^{-2}$ is used. The model is trained with batch size 64 and MAE as the loss function. Early stop is performed by monitoring the validation error.

FC-LSTM  The Encoder-decoder framework using LSTM with peephole [19]. Both the encoder and the decoder contain two recurrent layers. In each recurrent layer, there are 256 LSTM units, L1 weight decay is $2e^{-5}$, L2 weight decay $5e^{-4}$. The model is trained with batch size 64 and loss function MAE. The initial learning rate is $1e^{-4}$ and reduces to $1e^{-5}$ every 10 epochs starting from the 20th epochs. Early stop is performed by monitoring the validation error.

DCRNN  : Diffusion Convolutional Recurrent Neural Network. Both encoder and decoder contain two recurrent layers. In each recurrent layer, there are 64 units, the initial learning rate is $1e^{-2}$, and reduces to $1e^{-3}$ every 10 epochs starting at the 20th epoch and early stopping on the validation dataset is used. Besides, the maximum steps of random walks, i.e., $K$, is set to 3. For scheduled sampling, the thresholded inverse sigmoid function is used as the probability decay:

$$\epsilon_i = \frac{\tau}{\tau + \exp(i/\tau)}$$

where $i$ is the number of iterations while $\tau$ are parameters to control the speed of convergence. $\tau$ is set to 3,000 in the experiments.

E.1 Dataset

We conduct experiments on two real-world large-scale datasets:

- **METR-LA**  This traffic dataset contains traffic information collected from loop detectors in the highway of Los Angeles County [11]. We select 207 sensors and collect 4 months of data ranging from Mar 1st 2012 to Jun 30th 2012 for the experiment. The total number of observed traffic data points is 6,519,002.

- **PEMS-BAY**  This traffic dataset is collected by California Transportation Agencies (CalTrans) Performance Measurement System (PeMS). We select 325 sensors in the Bay Area and collect 6 months of data ranging from Jan 1st 2017 to May 31th 2017 for the experiment. The total number of observed traffic data points is 16,937,179.

The sensor distributions of both datasets are visualized in Figure 5.

In both of those datasets, we aggregate traffic speed readings into 5 minutes windows, and apply Z-Score normalization. 70% of data is used for training, 20% are used for testing while the remaining
Table 3: Performance comparison for DCRNN and GCRNN on the METRA-LA dataset.

<table>
<thead>
<tr>
<th></th>
<th>15 min</th>
<th></th>
<th>30 min</th>
<th></th>
<th>1 hour</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>RMSE</td>
<td>MAPE</td>
<td>MAE</td>
<td>RMSE</td>
<td>MAPE</td>
</tr>
<tr>
<td>DCRNN</td>
<td>2.77</td>
<td>5.38</td>
<td>7.5%</td>
<td>3.15</td>
<td>6.45</td>
<td>8.8%</td>
</tr>
<tr>
<td>GCRNN</td>
<td>2.80</td>
<td>5.51</td>
<td>7.5%</td>
<td>3.24</td>
<td>6.74</td>
<td>9.0%</td>
</tr>
</tbody>
</table>

10% for validation. To construct the sensor graph, we compute the pairwise road network distances between sensors and build the adjacency matrix using thresholded Gaussian kernel [18].

\[
W_{ij} = \exp \left( -\frac{\text{dist}(v_i, v_j)^2}{\sigma^2} \right) \quad \text{if} \quad \text{dist}(v_i, v_j) \leq \kappa, \quad \text{otherwise} \quad 0
\]

where \(W_{ij}\) represents the edge weight between sensor \(v_i\) and sensor \(v_j\), \(\text{dist}(v_i, v_j)\) denotes the road network distance from sensor \(v_i\) to sensor \(v_j\), \(\sigma\) is the standard deviation of distances and \(\kappa\) is the threshold.

E.2 Metrics

Suppose \(x = x_1, \ldots, x_n\) represents the ground truth, \(\hat{x} = \hat{x}_1, \ldots, \hat{x}_n\) represents the predicted values, and \(\Omega\) denotes the indices of observed samples, the metrics are defined as follows.

Root Mean Square Error (RMSE)

\[
\text{RMSE}(x, \hat{x}) = \sqrt{\frac{1}{|\Omega|} \sum_{i \in \Omega} (x_i - \hat{x}_i)^2}
\]

Mean Absolute Percentage Error (MAPE)

\[
\text{MAPE}(x, \hat{x}) = \frac{1}{|\Omega|} \sum_{i \in \Omega} \left| \frac{x_i - \hat{x}_i}{x_i} \right|
\]

Mean Absolute Error (MAE)

\[
\text{MAE}(x, \hat{x}) = \frac{1}{|\Omega|} \sum_{i \in \Omega} |x_i - \hat{x}_i|
\]

F More Experimental Results

F.1 Effect of spatial dependency modeling

To further investigate the effect of spatial dependency modeling, we compare DCRNN with the following variants: (1) DCRNN-NoConv, which ignores spatial dependency by replacing the random walk matrix with the identity matrix. This essentially means the forecasting of a sensor can be only be inferred from its historical readings; (2) DCRNN-UniConv, which only uses the forward random walk for diffusion convolution; Figure 6 shows the learning curves of these three models with roughly the same number of parameters. Without diffusion convolution, DCRNN-NoConv has much higher validation error. Moreover, DCRNN achieves the lowest validation error which shows the effectiveness of using bidirectional random walk. The intuition is that the bidirectional random walk gives the model the ability and flexibility to capture the influence from both the upstream and the downstream.

To investigate the effect of graph construction, we construct a undirected graph by setting \(\hat{W}_{ij} = \hat{W}_{ji} = \max(W_{ij}, W_{ji})\), where \(\hat{W}\) is the new symmetric weight matrix. Then we develop a variant of DCRNN denotes GCRNN, which uses the sequence to sequence learning with ChebNet graph convolution (Equation 3). Table 3 shows the comparison between DCRNN and GCRNN in METRA-LA dataset. DCRNN consistently outperforms GCRNN. The intuition is that directed graph better captures the asymmetric correlation between traffic sensors.
Figure 6: Learning curve for DCRNN and DCRNN without diffusion convolution. Removing diffusion convolution results in much higher validation error. Moreover, DCRNN with bi-directional random walk achieves the lowest validation error.

Figure 7: Effects of K and the number of units in each layer of DCRNN. K corresponds to the reception field width of the filter, and the number of units corresponds to the number of filters.

Figure 8: Performance comparison for different DCRNN variants. DCRNN, with the sequence to sequence framework and scheduled sampling, achieves the lowest MAE on the validation dataset. The advantage becomes more clear with the increase of the forecasting horizon.

Figure 9: Traffic time series forecasting visualization. DCRNN generates smooth prediction and is usually better at predict the start and end of peak hours.
Figure 10: Visualization of learned localized filters centered at different nodes with $K = 3$ on the METR-LA dataset. The star denotes the center, and the colors represent the weights. We can see (1) filters are localized around the center, and (2) the weights diffuse alongside the road network.

Figure 7 shows the effects of different parameters. $K$ roughly corresponds the reception field of the filter while the number of units corresponds to the number of filters. Larger $K$ enables the model to capture broader spatial dependency at the cost of increasing learning complexity. We observe that with the increase of $K$, the error on the validation dataset first quickly decrease, and then slightly increase. Similar behavior is observed for varying the number of units.

G Effect of temporal dependency modeling

To evaluate the effect temporal modeling including the sequence to sequence framework as well as the scheduled sampling mechanism, we further design four variants of DCRNN: (1) DCNN: in which we concatenate the historical observations as a fixed length vector and feed it into stacked diffusion convolutional layers to predict the future time series. We then train a single model for one step ahead prediction, and feed the previous prediction into the model as input to perform multiple steps ahead prediction. (2) DCRNN-SEQ: which uses the encoder-decoder sequence to sequence learning framework to perform multiple steps ahead forecasting. (3) DCRNN: similar to DCRNN-SEQ except for adding scheduled sampling.

Figure 8 shows the comparison of those four methods with regards to MAE for different forecasting horizons. We observe that: (1) DCRNN-SEQ outperforms DCNN by a large margin which conforms the importance of modeling temporal dependency. (2) DCRNN achieves the best result, and its superiority becomes more evident with the increase of the forecasting horizon. This is mainly because the model is trained to deal with its mistakes during multiple steps ahead prediction and thus suffers less from the problem of error propagation. We also train a model that always been fed its output as input for multiple steps ahead prediction. However, its performance is much worse than all the three variants which emphasizes the importance of scheduled sampling.

H Model Interpretation

To better understand the model, we visualize forecasting results as well as learned filters. Figure 9 shows the visualization of 1 hour ahead forecasting. We have the following observations: (1) DCRNN generates smooth prediction of the mean even when frequent oscillation exists in the traffic signal (as shown in Figure 9(a)). This reflects the robustness of the model. (2) DCRNN is more likely to accurately predict abrupt changes in the traffic speed than baseline methods (e.g., FC-LSTM). As shown in Figure 9(b), DCRNN is often able to predict the start and the end of the peak hours. This is because DCRNN captures the spatial dependency, and is able to utilize the speed changes in neighborhood sensors for more accurate forecasting. More visualizations are provided in Figure 11 and Figure 12. Figure 10 visualizes examples of learned filters centered at different nodes. The star denotes the center, and colors denote the weights. We can observe that (1) weights are well localized around the center, and (2) the weights diffuse based on road network distance.
Figure 11: Traffic time series forecasting visualization.
Figure 12: Traffic time series forecasting visualization.