EFFICIENT TRAFFIC FORECASTING WITH GRAPH EMBEDDING

PROJECT DETAILS ARE AVAILABLE AT HTTPS://GITHUB.COM/SYIN3/CS224W-TRAFFIC.

Jiahui Wang, Zhecheng Wang & Shuyi Yin
CS224W Analysis of Networks, Stanford University, Autumn 2018-2019
{jiahuiw,zhecheng,syin3}@stanford.edu

ABSTRACT

Traffic forecasting is critical for the planning and monitoring of modern urban systems. Traditional approaches rely on queuing theory, flow theory and simulation techniques; more recent data-driven methods fit time-series and latent space models. However, they all have limitations, largely due to unrealistic stationary assumptions, cumbersome parameter calibration, and failure to incorporate spatial and temporal aspects simultaneously. Therefore, any new promising method must address two challenges: (1) an efficient representation of the complex spatial dependency on road/ sensor networks; (2) non-linear temporal dynamics and long-term forecasting. In this project, we propose to model spatial dependency of the directed transportation network with graph embedding techniques, and to model the temporal dependency with RNN-based time series model. More specifically, our refined model captures both spatial and temporal information effectively and computational-efficiently, and achieves comparable accuracy with the state-of-the-art models in traffic forecasting.

1 INTRODUCTION

Transportation plays an important role in our daily life. With the increasing amount of vehicles in the urban area and the development of the autonomous vehicles operations, the need for long-term traffic forecasting grows rapidly. Accurate, efficient, and robust traffic forecasting model is essential for intelligent transportation systems (ITSs). Traditional methods for traffic forecasting are built on queuing theory and mathematical flow theory (Bellman, 1961; Drew, 1968), but they suffer from the curse of dimensionality. Data-driven methods emerged in recent years which apply time-series analysis and other statistic models, such as latent space models (Deng et al., 2016; Yu et al., 2016), while they fail to capture non-linear temporal dynamics of traffic flow.

Recently, machine learning, especially deep learning and neural networks have shown their power in many fields, and so as in traffic forecasting. These algorithms provide a new way to characterize the spatio-temporal dependencies in the long-term traffic forecasting (Li et al. (2018); Yu et al. (2018); Cheng et al. (2017)). With these methods, people can predict the traffic information such as speeds and volumes during a long period with acceptable accuracy. However, their models rely on complex architectures with large amount of parameters to capture both spatial and temporal dependencies and make the training and testing relatively slow, which hampers their application in real-world large-scale transportation network.

In this paper, we are going to explore the algorithms with data-driven approach to efficiently predict the future traffic speeds and volumes based on historical data. We seek to improve the previous models by increasing computation efficiency (Li et al. (2018); Yu et al. (2018); Cheng et al. (2017)) and balancing with accuracy. More specifically, we employ state-of-art graph embedding techniques, such as Node2Vec (Grover & Leskovec (2016)) and SDNE (Wang et al. (2016)), to capture spatial features of the network in advance, which have never been used for traffic flow forecasting before.

Incorporating the pre-calculated spatial features into the time-series model such as Gated Recurrent Unit (GRU), we expect our model to gain competing accuracy and better computing efficiency.

2 Related Work

One of the state-of-the-art modeling of traffic flow is introduced by Li et al. (2018). In their paper, Li et al. (2018) applied a *Diffusion Convolutional Recurrent Neural Network* (DCRNN) to capture both spatial and temporal dependencies. Their model, DCRNN, model the traffic flow as a diffusion process on a directed graph. The highlight of this paper is relating traffic flow to a diffusion process, which explicitly captures the stochastic nature of traffic dynamics. Despite the state-of-the-art accuracy of the work, the computational efficiency is poor due to the large model structures incorporating both Recurrent Neural Network (RNN) and Diffusion-Convolutional Neural Network (DCNN). Yu et al. (2018) introduces an efficient way to predict the traffic flow by using purely convolutional structure, which provides a way for processing large-scale networks. To encode the temporal dynamic behaviors of traffic flows, the paper applies convolutional operations on time series. However, this method can only be applied to undirected graphs for traffic forcasting. In reality, the traffic network is always two-way and may have different behaviors for in and out flows.

The papers we discuss above using sensors as nodes and sensor distances to derive edge weights. DeepTransport model (Cheng et al. (2017)) introduces another way, which forecasts the future traffic congestion level of a node by explicitly integrating the features from its upstream nodes and downstream nodes. However, beyond the scope of certain order, nodes are assumed to have no effect anymore. Therefore, the spatial features of the graph cannot be captured globally, which can potentially reduce the performance.

Graph embedding techniques can capture and encode spatial features of a graph (Belkin & Niyogi (2002), Shaw & Jebara (2009), Perozzi et al. (2014), Grover & Leskovec (2016), Bruna et al. (2013), Defferrard et al. (2016), Wang et al. (2016), Goyal & Ferrara (2018)). They can be divided into three categories: Factorization methods such as Laplacian eigenmaps (Belkin & Niyogi (2002)), Random-Walk-based methods such as DeepWalk (Grover & Leskovec (2016)) and Node2Vec (Grover & Leskovec (2016)), and deep-learning-based methods such as Graph Convolutional Network (Bruna et al. (2013), Defferrard et al. (2016)) and Structural Deep Network Embedding (SDNE) (Wang et al. (2016)). In general, deep-learning-based methods show stronger capability in capturing the inherent non-linear dynamics of the graph, and yields better representations of the networks (Goyal & Ferrara (2018)).

3 Methods

3.1 Problem Statement

We model a transportation sensor network as a graph $G(V, E, \mathbf{W})$, where V is the collection of all nodes, E is the collection of all edges, and $\mathbf{W}$ is the collection of edge weights. Each node corresponds to a sensor which measures several features at that location such as traffic speed and volume, and we use x_i^t to represent the traffic feature measurements of sensor (node) i at time t. Features of all nodes can be represented as $X^t \in \mathbb{R}^{N \times P}$, where P is the number of features for each node. Each edge represents the adjacency and relationship (up/downstream) between two nodes and the its weight characterize the road distance. Here we use the same weight construction scheme as in Li et al. (2018), which is called Thresholded Gaussian kernel (Shuman et al., 2013). For each entry in the graph adjacency matrix A, the weight of the edge from node v_i to node v_j, denoted as A_{ij}, is defined as:

$$A_{ij} = \begin{cases} \exp(-\frac{\text{dist}(v_i, v_j)^2}{\sigma^2}) & \text{if dist}(v_i, v_j) \leq k \\ 0 & \text{otherwise} \end{cases}$$

Where $dist(v_i, v_j)$ is the road network distance, rather than the cross-fly distance, from node v_i to node v_j. σ is the standard deviation of distances and k is the threshold. Based on the measurement of features (i.e. traffic speed and volume) at every nodes for S time steps in the past, $[X_{t-S+1}, X_{t-S+2}, ...X_t]$, our task is to develop a model to predict $[X_{t+1}, X_{t+2}, ...X_{t+T}]$ with the time horizon T in the future. The graph G is also taken as input.

3.2 SPATIAL FEATURE CAPTURING

Previous works (Li et al. (2018), Yu et al. (2016)) utilized graph convolution network-based models to characterize the spatial dependency. Their methods suffer from relatively low computation efficiency, which is a general drawback of models based on graph convolution network (GCN), since the amount of parameters to train in the GCN is huge. In this work, we aim to apply graph embedding to capture the spatial feature of the graph in advance, and then use it as a pre-calculated input in the time-series traffic prediction model. Such change separates the spatial and temporal dependency modeling and thus reduces the size and computation time of the time series prediction model, which enables efficient and scalable traffic forecasting. As far as we know, such separated spatial-temporal modeling with graph embedding has never been applied in traffic forecasting problems before.

Generally speaking, graph embedding, also named as node embedding within the scope of this paper, is to map each node into a vector to encode the spatial features of the graph. In this work, we use Node2Vec (Grover & Leskovec (2016)) and Structural Deep Network Embedding (SDNE) (Wang et al. (2016)), and compare them with Diffusion Convolutional Recurrent Neural Network (DCRNN) (Li et al. (2018)) which used graph convolution network as a graph embedding method.

3.2.1 NODE2VEC

Node2Vec (Grover & Leskovec (2016)) used random walk to preserves higher-order proximity between nodes. It maps each node to a low-dimensional space of features that maximizes the likelihood of preserving network neighborhoods of nodes, where the neighborhoods are defined by the occurrences of subsequent nodes in fixed length random walks. Dot product between two embedding vectors is used to represent the proximity between the corresponding nodes, and the target is to maximize the probability:

$$Pr(N_S(u)|f(u)) = \prod_{n_i \in N_S} \frac{exp(f(n_i) \cdot f(u))}{\sum_{v \in V} exp(f(v) \cdot f(u))}$$

$N_S(u)$ is the set of neighbours of u which is determined by the random walks starting from node u. Since comparing one node with all other nodes is very expensive, we approximate the denominator with negative sampling (Mikolov et al. (2013)). The random walks can be configured by two hyperparameters p and q to balance between capturing microscopic and macroscopic structural features. Increasing p encourages exploring far away and capturing macroscopic features, while increasing q encourages exploring locally and capturing microscopic features.

3.2.2 STRUCTURAL DEEP NETWORK EMBEDDING (SDNE)

SDNE (Wang et al. (2016)) utilizes deep autoencoders to preserve the network proximities in a semi-supervised manner. Autoencoder is an unsupervised machine learning model to encode features into latent variables with an "encoder" and reconstruct the feature vector from the latent variables with a "decoder". In SDNE, The input to the encoder is the adjacency matrix and the latent variables are the embedding vector, the output of the decoder is the reconstructed adjacency matrix. By minimizing the reconstruction error, the model is trained to recover the neighbourhoods from the embedding vectors, forcing the embedding vectors to preserve the 2nd order proximity. The loss function for this goal can be defined as:

$$L_{2nd} = ||(\hat{A} - A) \odot B||_2^2$$

Where $\hat{A}$ is the reconstructed adjacency matrix. B is the penalty weight matrix. $B_{ij} = \beta$ if $A_{ij} > 0$ and $B_{ij} = 1$ if $A_{ij} = 0$. $\odot$ denotes elementwise multiplication.

On the other hand, we also add a loss term to make similar nodes mapped closed with each other in the embedding space to preserve the 1st order proximity, which can regarded as the supervised information to constrain the the latent representations. The loss function for this goal can be defined as:

$$\begin{aligned} L_{1st} &= \sum_{i,j=1}^{n} A_{ij}||y_i - y_j||_2^2 \\ &= \sum_{i,j=1}^{n} \sum_{k=1}^{p} A_{ij}(Y_{ik}^2 + Y_{jk}^2) - 2 \sum_{i,j=1}^{n} \sum_{k=1}^{p} A_{ij} Y_{ik} Y_{jk} \\ &= 2tr(Y^T D Y) - 2tr(Y^T A Y) \\ &= 2tr(Y^T L Y) \end{aligned}$$

Where $Y = \{y_i\}_{i=1}^n$, and $L = D - A$ is the Laplacian matrix. The overall loss function is:

$$Loss = L_{2nd} + \alpha L_{1st} + \gamma L_{reg} = ||(\hat{A} - A) \odot B||_2^2 + 2\alpha tr(Y^T L Y) + \gamma L_{reg}$$

Where L_{reg} is the regularization loss.

3.2.3 Diffusion Convolutional Recurrent Neural Network (DCRNN)

Li et al. (2018) regards traffic flow as a diffusion process and uses diffusion convolutional layer to operate on the graph signal X. The operation of such layer is:

$$H_{:,q} = \mathbf{a}(\sum_{p=1}^{P} \sum_{k=0}^{K-1} (\theta_{k,q,1}(D_O^{-1}A)^k + \theta_{k,q,2}(D_I^{-1}A^T)^k) X_{:,p})$$

for $q \in \{1, ..., Q\}$, where P is the input dimension and Q is the output dimension, θ denotes trainable parameters, D_I and D_O are inward and outward diagonal degree matrix, respectively. **a** is the activation function. Such diffusion convolutional incorporate the features from K-hops neighbours to construct the hidden state of each node. By stacking several diffusion convolutional layers, the final output can embed the deep feature of the graph.

3.3 Temporal dependency modeling

To capture the temporal dependency of traffic networks, we apply sequence-to-sequence model (Sutskever et al. (2014)) with Gated Recurrent Units (GRU). The architecture is shown in Figure 1. Each GRU is the combination of the following operations:

$$\begin{aligned} r^t &= \sigma(f_{\Theta_r}([X^t, H^{t-1}]) + b_r) & u^t &= \sigma(f_{\Theta_u}([X^t, H^{t-1}]) + b_u) \\ C^t &= tanh(f_{\Theta_C}([X^t, r^t \odot H^{t-1}]) + b_C) & H^t &= u^t \odot H^{t-1} + (1 - u^t) \odot C^t \end{aligned}$$

$\odot$ denotes elementwise multiplication. σ is the activation function. For DCRNN, f is the diffusion convolutional layer with $[X^t, H^{t-1}]$ as input. In our work, we propose Fully Connected Recurrent Neural Network (FCRNN) with graph embedding techniques including both Node2Vec and SDNE. They are denoted as **FCRNN-n2v** and **FCRNN-SDNE**, respectively. For these two models, The input to the GRU is $X_{all}^t = [X^t, X_{embedding}]$. And $f_\Theta([X_{all}^t, H^{t-1}]) = \Theta_1 \cdot X_{all}^t + \Theta_2 \cdot H^{t-1}$,

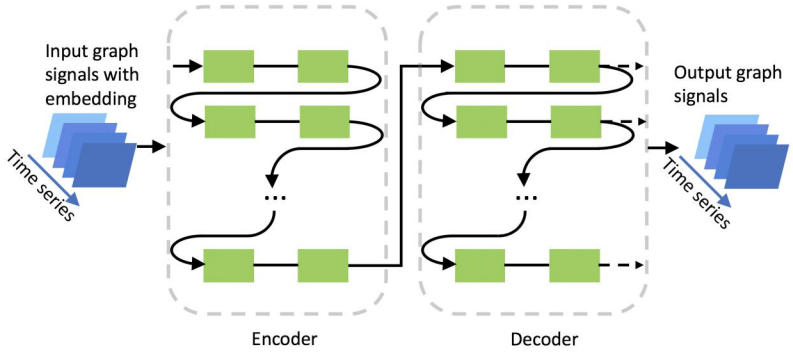


Figure 1: Architecture of Fully Connected Recurrent Neural Network (FCRNN) with embedding designed for spatiotemporal traffic forecasting. Each green box denotes a GRU module. The input is the graph features including embedding features (e.g. Node2Vec) for a time series. The encoder consists of 12 time steps, each step contains two GRU cells. The decoder has similar structure with the encoder and makes predictions based on either previous ground truth (during training) or the model output.

where Θ_1 and Θ_2 are trainable weights. The overall schematic of the sequence-to-sequence model is shown in Fig.1, including an encoder to receive S historical time steps of graph features, and a decoder to output the predictions of T future time steps of graph features. Here $S = T = 12$. Same scheduled sampling scheme is used for training as in Li et al. (2018).

3.4 Dataset

We use **METR-LA** dataset of LA county road network (by highway sensor stations), which is collected by Los Angeles Metropolitan Transportation Authority (LA-Metro). It contains both spatial and temporal aspects of the traffic. Li et al. (2018) selected 207 sensors in the METR-LA network to construct the graph and extracted the traffic measurement of all these sensors. We use the same dataset as theirs which has already been pre-processed by the authors of the paper. The network topology is visualized in Figure 2.

We load the graph published on Li's GitHub pages of Li et al. (2018). The weights and directedness are defined in section 3.1. Li et al. (2018) also aggregated sensor data into 5-minute intervals, so we have traffic speed reading for each of 207 sensors selected from March 1, 2012 to June 27, 2012, in a 5-minute interval. The traffic speed matrix is therefore of shape $(34272, 207)$. We further split our dataset into three parts: training, validation, test sets, with 70% for training, 20% for testing while the remaining 10% is for validation.

4 Results

With the traffic data and graph structures, we have trained three models:

- **FCRNN-n2v-dim-p**, where we use Node2Vec for graph embedding and choose hyperparameter **dim** (embedding dimension) and p in a grid search method.
- **FCRNN-sdne-dim-α**, where we use SDNE for graph embedding and **dim** and α are hyperparameters to be searched.
- **FCRNN-baseline** without any embedding attached to the original input feature matrix.

The training and testing processes are programmed in TensorFlow (Abadi et al., 2015) and conducted on AWS GPU instance (NVIDIA TESLA K80). Our reference model is **DCRNN** proposed in Li et al. (2018).

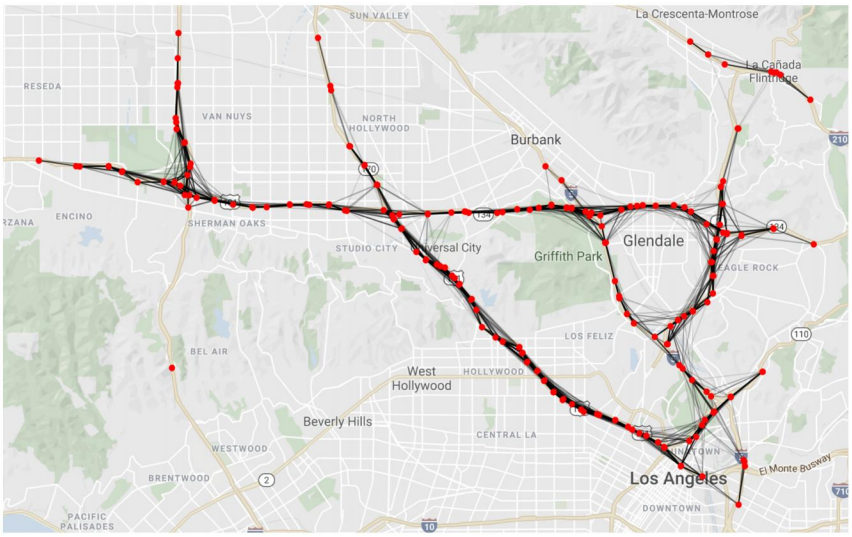


Figure 2: Visualization of the METR-LA traffic sensor network. The red dot denotes the node/sensor. The darkness of the edge represents the edge weight (The darker the edge, the higher the weight). Notice that as we defined, edges only exist for pair of nodes who are close enough in terms of road distance. There are disconnected parts of the graph.

4.1 Evaluation

We use three metrics to evaluate our model. In the following equations, $\mathbf{x}$ is the ground truth traffic speed vector and $\hat{\mathbf{x}}$ is the predicted value. N is the number of examples we are using. N may take values of $N_{\textbf{train}}$, $N_{\textbf{validation}}$ or $N_{\textbf{test}}$, depending on which set the metrics are applied to.

- Root Mean Square Error (RMSE): $\mathrm{RMSE}(\mathbf{x}, \hat{\mathbf{x}}) = \sqrt{\frac{1}{N}\sum_i^N (\mathbf{x} - \hat{\mathbf{x}})^2}$
- Mean Absolute Percentage Error (MAPE): $\mathrm{MAPE}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_i^N \mid \frac{x_i - \hat{x}_i}{x_i} \mid$
- Mean Absolute Error (MAE): $\mathrm{MAE}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N}\sum_i^N \mid x_i - \hat{x}_i \mid$

We also did a grid search on hyperparameters. As the traffic measurement matrix has its dimension of 2 (speed and corresponding time), we chose dim $= 2, 6, 14, 30, 62$ for our node embedding dimensions to make the last dimension of input feature matrix in the form of power of two. The chosen values for p in Node2Vec are $0.1, 0.3, 1.0, 3.0, 10.0$, while α for SDNE are tested on $1, 10, 100$. Therefore, we conducted $5 \times 5 + (5-1) \times 3 = 37$ experiments in total (we did not do dim $= 2$ case for SDNE).

4.2 Performance discussion

Table 1 shows the comparison of different approaches for 15 minutes, 30 minutes and 1 hour ahead forecasting on METR-LA test set. We observe the following phenomena:

- Best **FCRNN-n2v** and best **FCRNN-sdne** are better than **FCRNN-baseline**. This validates the overall strength that **node embedding** brings to this traffic forecasting problem. Spatial relationship between different node captured by node embedding locations contributes to better prediction of traffic flow.
- **FCRNN-n2v** and **FCRNN-sdne** are still outperformed by **DCRNN**. This indicate that capturing spatial and temporal dependencies together can achieve better performance than our methods which capture spatial dependence in advance. Our approach of incorporating node embeddings is still sub-optimal. Maybe we should figure out more sophisticated ways to

	15 min			30 min			1 hour		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
DCRNN	**2.77**	**5.38**	**7.3%**	**3.15**	**6.45**	**8.8%**	**3.60**	**7.60**	**10.5%**
FCRNN-n2v-14-0.1	**2.87**	**5.72**	**7.7%**	**3.36**	**6.96**	**9.6%**	**3.93**	**8.28**	**12.0%**
FCRNN-sdne-14-1	**2.86**	**5.71**	**7.7%**	**3.32**	**6.93**	**9.5%**	**3.87**	**8.23**	**11.8%**
FCRNN	2.99	5.91	7.9%	3.60	7.30	10.3%	4.47	8.99	13.8%

Table 1: Performance comparison for **DCRNN**, **FCRNN-n2v-14-0.1** and **FCRNN-sdne-14-1** on the METRA-LA dataset (test set) at different forecasting horizons. Notice **FCRNN-n2v-14-0.1** is the best among all **FCRNN-n2v** models and **FCRNN-sdne-14-1** is the best among all **FCRNN-sdne** models.

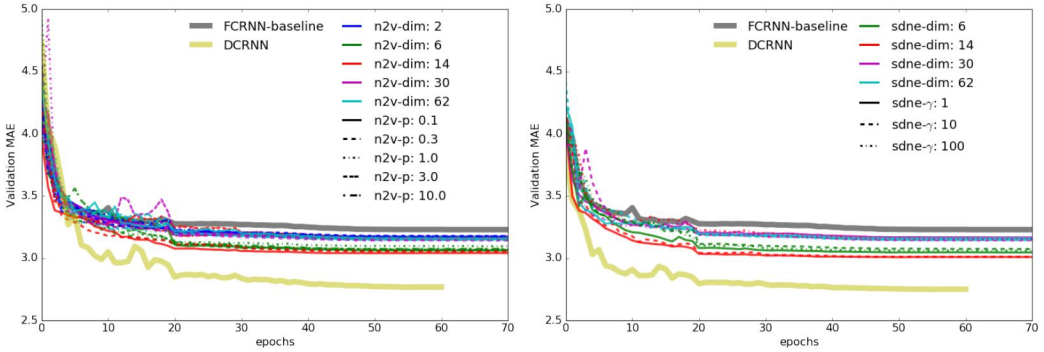
integrate the feature matrix produced by node embeddings to the time-series forecasting, instead of simply attaching them to the input feature matrix.

- Gaps widen with the increment of forecasting horizon. This is because the spatial-temporal dependency becomes increasingly non-linear with the growth of the horizon. The widening gap indicates our neural network is not yet capturing as good nonlinear spatial-temporal information as what **DCRNN** is capturing.

In Figure 3, we present the improvement of MAE on validation set along training. Specifically, we compared **FCRNN-n2v** series and **FCRNN-sdne** series with **DCRNN** in Figure 3(a) and Figure 3(b), respectively. Both series perform better than **FCRNN-baseline** but worse than **DCRNN**. **dim=**14 is the optimal dimension of embeddings for both **FCRNN-n2v** and **FCRNN-sdne**. An interesting finding is that for **FCRNN-n2v**, the best p value is 0.1, indicating that local and microscopic spatial features are more useful in traffic modeling than macroscopic features, as smaller p biases towards BFS-like exploration to capture the local structures. The best model among all **FCRNN-n2v** and **FCRNN-sdne** models is **FCRNN-sdne-14-1** (see in Figure 3(c)), which indicates that SDNE, as a deep-learning-based node embedding method, is better than Node2Vec in capturing the spatial features for traffic forecasting. This may be due to that deep neural network has stronger capability in capturing and representing the highly non-linear structural features of the graph.

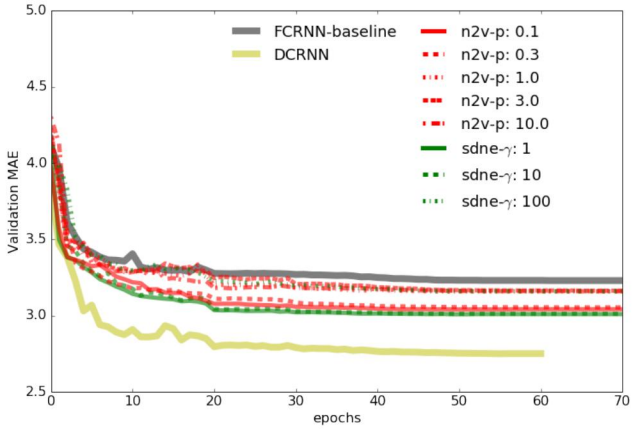
Despite the gap between the prediction accuracy of our models and the state-of-the-art accuracy, we would like to highlight the significant strength of our models on computation efficiency. The significance of this speeding-up is two-fold. First of all, in Figure 4, we compare the average training time per epoch on **NVIDIA Tesla K80** GPU. The training speed is almost consistent with varying dimensions of node embeddings. Clearly, **FCRNN-n2v** and **FCRNN-sdne** models are a lot faster to train than DCRNN: they consume 90% less time than the DCRNN model. Therefore, we have achieved nearly an order of magnitude speeding-up by using **FCRNN-n2v** or **FCRNN-sdne** models, at a price of tolerable decreasing in prediction accuracy. Secondly, since the computation efficiency of our models are much better than that of **DCRNN**, the real-time deployment of our models to support downstream task such as traffic signal control are much more viable. For real-world large-scale transportation networks, the advantage of our models on computation efficiency become even more obvious, since for our models, after we obtain the node embedding matrix with fixed dimension, the computation time needed for time-series forecasting only increases *linearly* with the number of the nodes N. By contrast, for **DCRNN**, the computation time needed for time-series forecasting increases *polynomially* with the number of the nodes N, since the $N \times N$ adjacency matrix is involved in the diffusion convolution operations.

To better understand the models, we visualize an example of forecasting results in Figure 5, which shows the ground truth and predicted traffic speed of 15 minutes ahead at sensor 101, on March 16, 2012. We have the following observations: (1) **FCRNN-n2v** and **FCRNN-sdne** both generate smooth prediction when small oscillation exists in the traffic speeds. This reflects the robustness of



(a) Comparing training process of **FCRNN-n2v-dim-p** series to **DCRNN**, measured by validation set MAE. Best performance is obtained at dim $= 14$.

(b) Comparing training performances of the **FCRNN-sdne-dim-α** series to **DCRNN**, by validation set MAE. Best performance is obtained at dim $= 14$.



(c) Comparing training performances of models of the best dimension dim $= 14$ in two series **FCRNN-n2v-dim-p** and **FCRNN-sdne-dim-α** to **DCRNN**, on **validation set MAE**. Bests of the best models are **FCRNN-n2v-14-0.1** and **FCRNN-sdne-14-1**, which are very close.

Figure 3: We can see all models stablize after about 40 epochs. All newly proposed models are between the boundaries: **DCRNN** and **FCRNN-baseline**. Best dimensions are 14 in both cases. More specfically, **FCRNN-n2v-14-0.1** and **FCRNN-sdne-14-1** are best of the bests. But notice, some other hyperparameters are still not optimized, such as *number of RNN layers*, *number of RNN cells in each layer*.

FCRNN-n2v and **FCRNN-sdne** models. (2) **FCRNN-n2v** and **FCRNN-sdne** are more likely to predict abrupt changes than **DCRNN**, which shows that our models may be better in predicting the accidental cases of traffic such as abrupt traffic congestions and traffic accidents. (3) There seems to be a lag of predicted speed for all these four models, compared to true traffic recordings. The reason underlying the lag is an open question and it is an interesting topic for our future work.

5 Conclusions and Future Works

In this work, we have developed machine-learning-based models to capture the spatial-temporal dependencies of a dynamic traffic network utilizing graph embedding techniques and thus make traffic forecasting. Compared to previous models, in our work the spatial features are captured in advance, which is separated from temporal modeling. Based on the evaluation metrics, our **FCRNN-n2v** and **FCRNN-sdne** models perform better than the baseline model **FCRNN-baseline** but the accuracy performance is lower than but closed to the **DCRNN**. Inclusion of node embeddings to capture

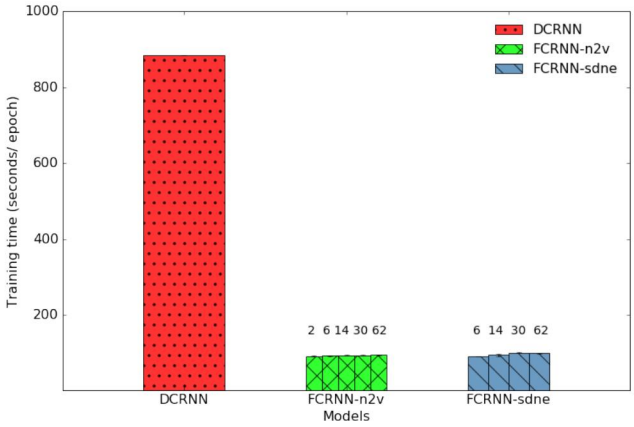


Figure 4: Comparing training time of two series of **FCRNN** models with **DCRNN**. The numbers on top of the bars denote the embedding dimensions of the models.

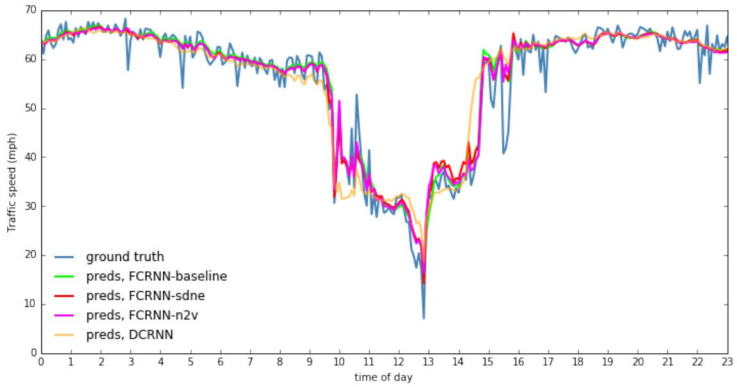


Figure 5: Visualizing predictions of four models at Sensor #101, on the 16^{th} day of data recorded. We can see all four models can predict fairly well on the true traffic speed (mph).

spatial features of the graph is fruitful. However, for long-term forecasting, the **DCRNN** model performs better. Hyper-parameters tuning produces two best models: **FCRNN-n2v-14-0.1** with embedding dimension of 14 and p value of 0.1, and **FCRNN-sdne-14-1** with embedding dimension of 14 and α value of 1, with the latter being an even better one. For **FCRNN-n2v**, smaller p yields better prediction accuracy, indicating that local features captured by BFS-like exploration is more useful for traffic modeling than macroscopic features, as the traffic forecasting of one node mainly relies on the information from nearby nodes. Despite the lower prediction accuracy than **DCRNN**, our two models achieve much better computation efficiency, namely $10\times$ faster than the **DCRNN** model. This work therefore is of great value because our **FCRNN-n2v** and **FCRNN-sdne** models have comparable accuracy performances with **DCRNN**, but save tremendous computation time and power, which makes the deployment in large-scale traffic networks much more viable. In the future, we may think of a more sophisticated way to incorporate static spatial embedding features to the dynamic time-series models to improve the prediction accuracy.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pp. 585–591, 2002.

Richard E Bellman. *Adaptive control processes: a guided tour*, volume 2045. Princeton university press, 1961.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Xingyi Cheng, Ruiqing Zhang, Jie Zhou, and Wei Xu. Deeptransport: Learning spatial-temporal dependency for traffic condition forecasting. *arXiv preprint arXiv:1709.09585*, 2017.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, Linhong Zhu, Rose Yu, and Yan Liu. Latent space model for road networks to predict time-varying traffic. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1525–1534. ACM, 2016.

Donald R Drew. Traffic flow theory and control. institution, 1968.

Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864. ACM, 2016.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=SJiHXGWAZ`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710. ACM, 2014.

Blake Shaw and Tony Jebara. Structure preserving embedding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 937–944. ACM, 2009.

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. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234. ACM, 2016.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*, 2018.

Hsiang-Fu Yu, Nikhil Rao, and Inderjit S Dhillon. Temporal regularized matrix factorization for high-dimensional time series prediction. In *Advances in neural information processing systems*, pp. 847–855, 2016.