Graph Attention Recurrent Neural Networks for Correlated Time Series Forecasting

Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang
Department of Computer Science, Aalborg University, Denmark
{razvan,cguo,byang}@cs.aau.dk

ABSTRACT

We consider a setting where multiple entities interact with each other over time and the time-varying statuses of the entities are represented as multiple *correlated time series*. For example, speed sensors are deployed in different locations in a road network, where the speed of a specific location across time is captured by the corresponding sensor as a time series, resulting in multiple speed time series from different locations, which are often correlated. To enable accurate forecasting on correlated time series, we proposes *graph attention recurrent neural networks*. First, we build a graph among different entities by taking into account spatial proximity and employ a multi-head attention mechanism to derive adaptive weight matrices for the graph to capture the correlations among vertices (e.g., speeds at different locations) at different timestamps. Second, we employ recurrent neural networks to take into account temporal dependency while taking into account the adaptive weight matrices learned from the first step to consider the correlations among time series. Experiments on a large real-world speed time series data set suggest that the proposed method is effective and outperforms the state-of-the-art in most settings.

ACM Reference Format:
Razvan-Gabriel Cirstea, Chenjuan Guo, Bin Yang. 2018. Graph Attention Recurrent Neural Networks for Correlated Time Series Forecasting. In *MileTS '19: 5th KDD Workshop on Mining and Learning from Time Series, August 5th, 2019, Anchorage, Alaska, USA*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Complex cyber-physical systems (CPSs) often consist of multiple entities that interact with each other across time. With the continued digitization, various sensor technologies are deployed to record time-varying attributes of such entities, thus producing correlated time series [5]. One representative example of such CPSs is road transportation system [6, 20], where the speeds on different roads are captured by, e.g., loop detectors and speed cameras, as multiple speed time series [9, 10].

Accurate forecasting of correlated time series have the potential to reveal holistic system dynamics of the underlying CPSs, including predicting future behavior [5] and detecting anomalies [12, 13], which are important to enable effective operations of the CPSs. For example, in an intelligent transportation system, analyzing speed time series enables travel time forecasting, early warning of congestion, and predicting the effect of incidents, which help drivers make routing decisions [7, 15]. To enable accurate and robust correlated time series forecasting, it is essential to model the spatio-temporal correlations among multiple time series. To this end, we propose *graph attention recurrent neural networks (GA-RNNs)*.

We first build a graph among different entities by taking into account spatial proximity. In the graph, vertices represent entities and two vertices are connected by an edge if the two corresponding entities are nearby. After building the graph, we apply multi-head attention to learn a weight matrix. For each vertex, the weight matrix indicates, among all the vertex's neighbor vertices, which neighboring vertices' speeds are more relevant when predicting the speed of the vertex.

Next, since recurrent neural networks (RNNs) are able to well capture temporal dependency, we modify classic RNNs to capture spatio-temporal dependency. Specifically, we replace weight multiplications in classic RNNs with convolutions that take into account graph topology (e.g., graph convolution or diffusion convolution [14]). However, instead of using a static adjacency matrix, we employ the attention weight matrix learned from the first step to obtain adaptive adjacency matrices. Here, with the learned attention weight matrices and the inputs at different timestamps, we obtain different adjacency matrices at different timestamps, which are able to capture the dynamic correlations among different time series at different timestamps.

To the best of our knowledge, this is the first study that utilize attention to derive adaptive adjacency matrices which are then utilized in RNNs to capture spatio-temporal correlations among time series to enable accurate forecasting for correlated time series. Experiments on a large real-word traffic time series offer evidence that the proposed method is accurate and robust.

2 PROBLEM DEFINITION

Consider a cyber-physical system (CPS) where we have N entities. The status of the i-th entity across time (e.g., from timestamps 1 to t) is represented by a time series $TS^{(i)} = \langle x_1^{(i)}, x_2^{(i)}, \ldots, x_t^{(i)} \rangle$, where a K-dimensional vector $x_j^{(i)}$ records K features (e.g., speed and traffic flow) of the i-th entity at timestamp j.

Given historical statuses of all entities, we aim at predicting the future statuses of all entities. More specifically, we have historical statuses covering a window $[t_{a+1}, t_{a+l}]$ that contains l time stamps, and we aim at predicting the future statuses in a future window $[t_{a+l+1}, t_{a+l+p}]$ that contains p time stamps. We call this problem p-step ahead forecasting.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MileTS '19, August 5th, 2019, Anchorage, Alaska, USA
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

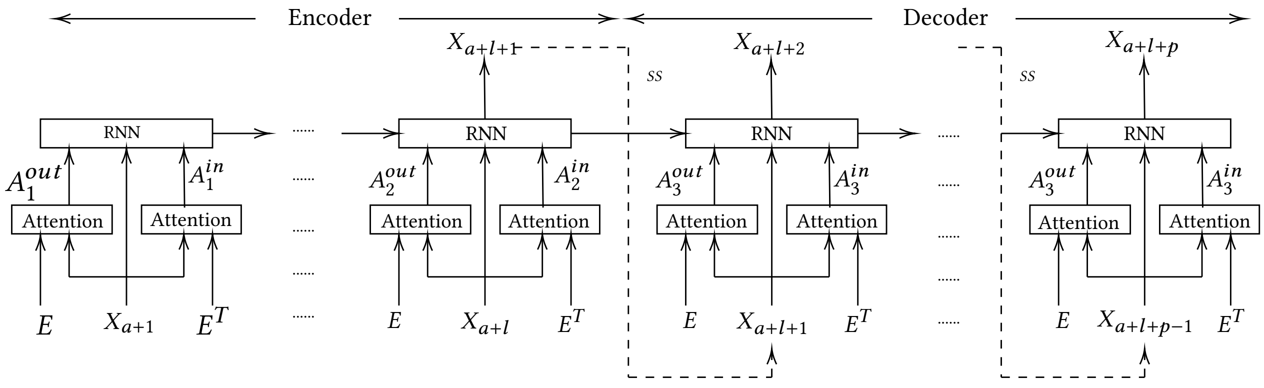

Figure 1: Graph Attention Recurrent Neural Network

3 GRAPH ATTENTION RNNS

We proceed to describe the proposed graph attention recurrent neural network (*GARNN*) to solve the *p*-step ahead forecasting.

3.1 Graph Signals

We first build a directed graph $G = (V, E)$ where each vertex $v \in V$ represents an entity in the CPS, which is often associated with spatial information such as longitude and latitude. Since we have N entities in total, we have $|V| = N$. Edges are represented by adjacency matrix $E \in \mathbb{R}^{NxN}$, where $E[i, j] = 1$ represents an edge from the i-th to the j-th vertices. If the entities are already embedded into a spatial network, e.g., camera sensors deployed in road intersections in a road network, we connect two entities by an edge if they are connected in the spatial network. Otherwise, we connect two entities by an edge if the distance between them is small [14].

At each time stamp t, each entity is associated with K features (e.g., speed, flow). We introduce a graph signal $X_t \in \mathbb{R}^{NxK} = [x_t^{(1)}, x_t^{(2)}, \ldots, x_t^{(N)}]^T$ to represent all features from all entities at timestamp t. Based on the concept of graph signals, the problem becomes learning a function that takes as input l past graph signals and outputs p future graph signals:

$$\langle X_{a+1}, ..., X_{a+l} \rangle \to \langle \hat{X}_{a+l+1}, ..., \hat{X}_{a+l+p} \rangle.$$

3.2 GARNN Framework

The proposed GARNN consists of two parts, an attention part and an RNN part, which follow an encoder-decoder architecture as shown in Fig. 1. At each timestamp, we firstly model the spatial correlations among different entities using multi-head attention as two attention weight matrices A_t^{out} and A_t^{in}, which consider the outgoing and incoming traffic, respectively. Next, the two attention weight matrices are fed into an RNN unit together with the input graph signal at the time stamp, which facilitate the RNN unit not only capture the temporal dependency but also the spatial correlations, making it is possible to capture the spatio-temporal correlations among different time series.

3.3 Spatial Modeling

To capture the spatial correlations among different entities at a specific timestamp, we employ attention mechanism [2]. The idea is to determine, in order to predict the features of an entity, i.e., a vertex, how much should we consider the features of the vertex's neighbour vertices. We may consider different neighbour vertices differently at different timestamps. Specifically, for each vertex i, we compute an attention score w.r.t. to each vertex in $NB(i) = \{j|E[i, j] = 1\} \cup \{i\}$, i.e., vertex i's neighbour vertices and itself.

We proceed to show how to compute an attention score for vertex i. Recall that for any vertex i in the graph, its features at timestamp t is represented by a K-dimensional vector $x_t^{(i)} \in \mathbb{R}^K$. Vertex j is a vertex from $NB(i)$. Attention score $A_t[i, j]$ indicates how much attention should be paid to vertex j's features in order to estimate vertex i's features at timestamp t, which is computed based on Equation 1, where $[\cdot||\cdot]$ represents the concatenation operator.

$$A_t[i, j] = \frac{exp(LReLU(v^\intercal [Wx_t^{(i)} \mid\mid Wx_t^{(j)}]))}{\sum_{m \in NB(i)} exp(LReLU(v^\intercal [Wx_t^{(i)} \mid\mid Wx_t^{(m)}]))} \quad (1)$$

First, we embed the features from each vertex with an embedding matrix $W \in \mathcal{R}^{F \times K}$, where F is the size of the embedding. Then, we concatenate the embeddings of the two vertices' features $Wx_t^{(i)}$ and $Wx_t^{(j)}$, which is fed into an attention network. The attention network is constructed as a single-layer feed forward neural network, parameterized by a weight matrix $v \in \mathbb{R}^{2 \cdot F \times 1}$. In addition, we apply LeakyReLU (with a negative input slope of 0.2) as activation function (denoted as *LReLU* in Equation 1). Finally, we apply softmax to normalize the final output to obtain $A_t[i, j]$.

Motivated by [17], we observe that stacking multiple attention networks, a.k.a., attention heads, is beneficial since each attention network can specialise on capturing different interactions. Assume that we use a total of C attention networks, each network has its own embedding matrix $W^{(c)} \in \mathcal{R}^{F \times K}$ and weight matrix $v^{(c)} \in \mathbb{R}^{2 \cdot F \times 1}$ in the feed forward neural network. There are multiple ways of combining the attention heads, in our experiments we used average according to Equation 2,

$$A_t[i, j] = \frac{1}{C} \sum_{c=1}^{C} \frac{exp(LReLU(v_c^\intercal [W_c x_t^{(i)} \mid\mid W_c x_t^{(j)}]))}{\sum_{m \in NB(i)} exp(LReLU(v_c^\intercal [W_c x_t^{(i)} \mid\mid W_c x_t^{(j)}]))} \quad (2)$$

So far, for each vertex, we have computed the attentions to all its neighbors, which captures the influence of the outgoing traffic. On the other hand, it is also of interest to capture the influence of the incoming traffic. To this end, we use the transpose of the adjacency matrix E to define neighboring vertices and apply the same attention mechanism to obtain another attention matrix to represent the influence of incoming traffic. Finally, we obtain two attention matrices $A_t^{out} \in \mathbb{R}^{N \times N}$ and $A_t^{in} \in \mathbb{R}^{N \times N}$, which capture the influence from outgoing traffic and incoming traffic, respectively.

3.4 Spatio-Temporal Modeling

We integrate the learned attention matrices into classical recurrent neural networks to capture spatio-temporal correlations. We follow an encoder-decoder architecture as shown in Figure 1, which is able to well capture the temporal dependencies across time. Next, we replace the matrix multiplications in RNN units by convolutions that take into account graph topology such as diffusion convolution or graph convolution. Here, instead of using a static adjacency matrix E that only captures connectivity, the convolution here employs the learned attention matrices A_t^{out} and A_t^{out} at timestamp t as adaptive adjacency matrices since at different time stamps, we obtain different attention matrices. We proceed to define a diffusion convolution operator $\otimes$ on a graph signal $X_t \in \mathcal{R}^{NxK}$ using the learned attention matrices A_t^{out} and A_t^{in} at timestamp t in Equation 3.

$$X_t \otimes \Theta = \alpha(\sum_{k=1}^{K} \sum_{h=1}^{H} (\theta_{k,h,1}(A_t^{out})^h + \theta_{k,h,2}(A_t^{in})^h) X_t[\cdot, k]) \qquad (3)$$

Here, α is an activation function, matrix $\Theta \in \mathcal{R}^{K \times H \times 2}$ is a filter to be learned, and $X_t[\cdot, k]$ represents the k-th column of graph signal X_t, which is the k-th features of all entities. We often apply Q different filters to perform diffusion convolutions and then concatenate the results into a matrix $X_t \otimes \Theta = [X_t \otimes \Theta_1 || X_t \otimes \Theta_2 || \dots || X_t \otimes \Theta_Q]$. Finally, graph signal X_t is convoluted into matrix $X_t \otimes \Theta \in \mathcal{R}^{N \times Q}$.

Next, we integrate the proposed graph attention based convolution into an RNN. Here, we use a Gated Recurrent Unit (GRU) [4] as an RNN unit to illustrate the integration, where the matrix multiplications in classic GRU are replaced by the graph attention based diffusion convolutional process as defined in Eq. 3.

$$\begin{aligned} r_t &= \sigma(\Theta_r \otimes [X_t || H_{t-1}] + b_r) \\ u_t &= \sigma(\Theta_u \otimes [X_t || H_{t-1}] + b_u) \\ \hat{h}_t &= tanh(\Theta_{\hat{h}} \otimes [X_t || (r_t \odot H_{t-1})] + b_{\hat{h}}) \\ H_t &= u_t \odot H_{t-1} + (1 - u_t) \odot \hat{h}_t \end{aligned}$$

Here, $X_t \in \mathcal{R}^{N \times K}$ is the graph signal at timestamp t, H_t is the output at timestamp t. r_t, u_t and $\hat{h}_t$ represent the reset gate, update gate, and candidate context at time t. $\otimes$ indicates the proposed graph attention based diffusion convolution, and Θ_r, Θ_u, and $\Theta_{\hat{h}}$ are the filters used in the three convolutions. $\odot$ is Hadamard product.

The proposed graph attention is generic in the sense that it provides a data-driven manner to produce adaptive adjacency matrices and thus can be integrated with different kinds of convolutions that utilize graph adjacency matrices. Such convolutions often use a static adjacency matrix, while the proposed graph attention allows us to employ adaptive adjacency matrices at different timestamps, which are expected to better capture the spatio-temporal correlations among different time series.

4 EMPIRICAL STUDY

Experimental Setup: We conducted experiments on a large real world traffic dataset METR-LA from [14]. The dataset consists of speed measurements from 207 loop detectors spread across Los Angeles highways. The data was collected between March 1st 2012 and June 30th 2012 with a frequency of every 5 minutes.

We follow the same experimental setup as [14]. We build a graph by connecting from sensors i to j if the road network distance from i to j is small [14]. Since road network distance is used, the distance from i to j may be different from the distance from j to i, making the adjacency matrix E asymmetric. We use 70% of the data for training, 10% for validation and 20% for testing. We consider three metrics to evaluate the prediction accuracy: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE). We consider a setting where we use $l = 12$ past observations to predict the next $p = 12$ steps ahead and report the errors at three different intervals (15, 30, and 60 mins).

Implementation Details: The method is implemented in Python 3.6 using Tensorflow 1.7. A server with Intel Xeon Platinum 8168 CPU and 2 Tesla V100 GPUS are used to conduct all experiments. We trained the model using Adam optimizer with 0.01 as learning rate which decreases every 10 epochs after the 40th iteration. We used a total of 3 attention heads with an embedding size of 16. In addition we used a 2 layer GRU, with 64 units, a batch size of 64, and the same scheduled sampling technique described in [14].

Experimental Results: We consider the following two baseline algorithms from [14]: graph convolutional recurrent neural network (GCRNN) and diffusion convolutional recurrent neural network (DCRNN). Since GCRNN and DCRNN employ graph based convolutions, where GCRNN uses graph convolution and DCRNN uses diffusion convolution [14], we incorporate the proposed graph attention based adaptive adjacency matrix into the two methods to obtain GA-GCRNN and GA-DCRNN.

Accuracy: We compare the proposed GA-GCRNN and GA-DCRNN with the two baselines in Table 1. Since we use exactly the same experiment setup, all the results of the baselines are taken directly from [14] (written in italics in Table 1). A more in-dept description of the baselines, along with hyper-parameters can be found in [14].

T	Metric	*GCRNN*	GA-GCRNN	*DCRNN*	GA-DCRNN
15 min	MAE	2.80	2.76	2.77	**2.75**
	RMSE	5.51	5.33	5.38	**5.28**
	MAPE	7.5%	7.1%	7.3%	**7.0%**
30 min	MAE	3.24	3.21	**3.15**	3.19
	RMSE	6.74	6.45	6.45	**6.39**
	MAPE	9.0%	8.8%	8.8%	**8.0%**
1 hour	MAE	3.70	3.70	**3.6**	3.72
	RMSE	8.16	7.68	**7.59**	7.64
	MAPE	10.9%	10.9%	**10.5**%	11.0%
Avg.	MAE	3.28	3.22	**3.17**	3.22
	RMSE	6.80	6.48	6.47	**6.43**
	MAPE	9.13%	8.93%	8.86%	**8.66**%

Table 1: Evaluation of GARNNs

We first compare GA-GCRNN vs GCRNN, where GA-GCRNN outperforms GCRNN in all settings (see the underline values in Table 1). Next, GA-DCRNN outperforms DCRNN in most settings (see the bold values in Table 1), especially in short term predictions at 15 and 30 minutes. On average, GA-DCRNN is better when using RMSE and MAPE. This suggests that GA-DCRNN avoids having large prediction errors but may have more small prediction errors than does DCRNN. This shows that GA-DCRNN better captures the general trend of the underlying traffic data.

Efficiency: While DCRNN takes on average 271 s for one epoch, GA-DCRNN requires 401 s. The discrepancy between the two models is due to the attention mechanism, which requires more computation. Note that when implementing the attention we followed [18], but a more efficient way is available [16]. For the same reason, GA-GCRNN also takes longer time than GCRNN does.

Summary: Overall, the results suggest that the proposed graph head attention based adaptive adjacency matrices can be easily integrated with convolutions that consider graph topology and has a great potential to enable more accurate predictions, especially for relatively short term predictions. It is of interest to further investigate how to improve long term predictions.

5 RELATED WORK

For time series forcasting, auto-regressive models, e.g., ARIMA, is widely used as a baseline method. Hidden Markov models are also often used to enable time series forecasting. A so-called spatio-temporal HMM (STHMM) is able to consider spatio-temporal correlations among traffic time series from adjacent edges [21].

Neural networks are able to capture non-linear dependencies within the data, which enable non-linear forecasting models. In particular, Recurrent Neural Networks (RNNs) are used with successes in multiple domains such as traffic time series prediction [14] or wind forecasting [19], due to their recurrent nature that is able to capture temporal relationships within the data. Another well known type of deep learning models are Convolutions Neural Networks (CNNs). CNNs often work on grid-based inputs, limiting its applicability to graph-based inputs such as road networks. Recently, Bruna et al. [3] introduce Graph Convolutional Networks (GCNs) that combines spectral graphs theory with CNNs. A recent study employs GCNs to fill in missing values for uncertain traffic time series [8]. [1] proposes Diffusion Convolution Neural Networks (DCNNs) which fall under the non-spectral methods. DCNN works on the assumption that the further two nodes are in terms of graph topology, the lower impact they should have.

DCRNN [14] extends DCNN with RNN so that it also captures time dependencies within the data. Our work is closely related and builds on top of DCRNN. The main difference is that DCRNN assumes that the adjacency matrix used in random walks is static. However, this assumption might not always hold and we propose to learn an adaptive adjacency matrix at each time stamp using attention mechanism that considers graph topology. Multi-task learning [11] has also been applied for correlated time series forecasting, where both a CNN and a RNN are combined to both forecast future values and reconstruct historical values [5]. Our work resembles [18] and [22]. The main difference is that our model tries to learn a dynamic adjacency matrix which can afterwards be used with any type of graph-RNN like structure to update each node embedding by taking into account its neighbours offering more flexibility.

6 CONCLUSION

We propose a generic method to obtain adaptive adjacency matrices using graph attention which can be integrated seamlessly with existing graph based convolutions such as graph convolution and diffusion convolution. More specifically, we show how the integration of adaptive adjacency matrices and recurrent neural networks is able to improve the correlated time series predictions where the relationships among different time series can be captured as a graph. Preliminary experimental results show great potential when using adaptive adjacency matrices, especially for short term predictions.

REFERENCES

[1] James Atwood and Don Towsley. 2015. Search-Convolutional Neural Networks. *CoRR* abs/1511.02136 (2015). arXiv:1511.02136 http://arxiv.org/abs/1511.02136

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral Networks and Locally Connected Networks on Graphs. *CoRR* abs/1312.6203 (2013). arXiv:1312.6203 http://arxiv.org/abs/1312.6203

[4] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014). arXiv:1412.3555 http://arxiv.org/abs/1412.3555

[5] Razvan-Gabriel Cirstea, Darius-Valer Micu, Gabriel-Marcel Muresan, Chenjuan Guo, and Bin Yang. 2018. Correlated Time Series Forecasting using Multi-Task Deep Neural Networks. In *CIKM*. 1527–1530.

[6] Zhiming Ding, Bin Yang, Yuanying Chi, and Limin Guo. 2016. Enabling Smart Transportation Systems: A Parallel Spatio-Temporal Database Approach. *IEEE Trans. Computers* 65, 5 (2016), 1377–1391.

[7] Chenjuan Guo, Bin Yang, Jilin Hu, and Christian S. Jensen. 2018. Learning to Route with Sparse Trajectory Sets. In *ICDE*. 1073–1084.

[8] Jilin Hu, Chenjuan Guo, Bin Yang, and Christian S. Jensen. 2019. Stochastic Weight Completion for Road Networks using Graph Convolutional Networks. In *ICDE*. 1274–1285.

[9] Jilin Hu, Bin Yang, Chenjuan Guo, and Christian S. Jensen. 2018. Risk-aware path selection with time-varying, uncertain travel costs: a time series approach. *VLDB J.* 27, 2 (2018), 179–200.

[10] Jilin Hu, Bin Yang, Christian S. Jensen, and Yu Ma. 2017. Enabling time-dependent uncertain eco-weights for road networks. *GeoInformatica* 21, 1 (2017), 57–88.

[11] Tung Kieu, Bin Yang, Chenjuan Guo, and Christian S. Jensen. 2018. Distinguishing Trajectories from Different Drivers using Incompletely Labeled Trajectories. In *CIKM*. 863–872.

[12] Tung Kieu, Bin Yang, Chenjuan Guo, and Christian S. Jensen. 2019. Outlier Detection for Time Series with Recurrent Autoencoder Ensembles. In *IJCAI*.

[13] Tung Kieu, Bin Yang, and Christian S. Jensen. 2018. Outlier Detection for Multidimensional Time Series Using Deep Neural Networks. In *MDM*. 125–134.

[14] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *ICLR*.

[15] Huiping Liu, Cheqing Jin, Bin Yang, and Aoying Zhou. 2018. Finding Top-k Shortest Paths with Diversity. *IEEE Trans. Knowl. Data Eng.* 30, 3 (2018), 488–502.

[16] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. 2018. Geniepath: Graph neural networks with adaptive receptive paths. *arXiv preprint arXiv:1802.00910* (2018).

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* abs/1706.03762 (2017).

[18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[19] Arpita Yadav and Kapil Sahu. 2017. Wind forecasting using artificial neural networks: a survey and taxonomy. *International Journal of Research In Science & Engineering* 3 (2017).

[20] Bin Yang, Jian Dai, Chenjuan Guo, Christian S. Jensen, and Jilin Hu. 2018. PACE: a PAth-CEntric paradigm for stochastic path finding. *VLDB J.* 27, 2 (2018), 153–178.

[21] Bin Yang, Chenjuan Guo, and Christian S. Jensen. 2013. Travel Cost Inference from Sparse, Spatio-Temporally Correlated Time Series Using Markov Models. *PVLDB* 6, 9 (2013), 769–780.

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