Attention-based Graph Estimation and Directed Convolution for Prediction of Traffic Conditions

Michael P. Kenning
Department of Computer Science
Swansea, Wales, UK
788486@swansea.ac.uk

Xianghua Xie*
Department of Computer Science
Swansea, Wales, UK
x.xie@swansea.ac.uk

ABSTRACT

In this paper we present a novel attention-based graph estimation strategy to learn the graph structure of a road network. The strategy uses unique compositions cyclical traffic information to estimate and fuse long- and short-term graphs. We also introduce an attention-based convolution for directed graphs that models traffic information as separate flows into and out of vertices. The local attention mechanism in the directed attention-based convolution hence complements the global attention mechanism in the graph estimation. We evaluate these techniques using a modified Traffic Transformer on two commonly used datasets: METR-LA and PeMS-Bay. From the results we conclude that the graph estimation strategy and the convolutional layer lead to models that are robust to troublesome traffic conditions.

KEYWORDS

graph, deep learning, traffic prediction, graph estimation

ACM Reference Format:
Michael P. Kenning and Xianghua Xie. 2023. Attention-based Graph Estimation and Directed Convolution for Prediction of Traffic Conditions. In *Proceedings of 10th Workshop on Deep Learning on Graphs 2023 (KDD-DLG'23).* ACM, New York, NY, USA, 5 pages.

ACKNOWLEDGEMENT

The experiments were possible with the support of AccelerateAI, a supercomputer cluster provided by Supercomputing Wales and part-funded by the European Regional Development Fund through the Welsh Government.

1 INTRODUCTION

The graph has proved to be a useful mathematical object for representing irregular domains, as the flourishing research over the last decade has shown [3]. Graphs are used to structure the irregular relations between entities and convolutional layers based on graphs have seen broad application, such as to electroencephalographys [11] and citation networks [5].

The primary concern in applying graph techniques to new domains is to design a graph appropriate to the domain. A common construction for traffic datasets, for example, uses a thresholded Gaussian kernel [10] of the distances between traffic sensors, an approach which recurs in the research [1, 7, 8, 12]. The traffic sensors do not necessarily relate to one another symmetrically. That the traffic detected by a sensor located on a major road is important to predicting traffic on a minor road does not imply that the traffic on a minor road is as important in predicting traffic on the major road. These relations may be represented as directed graphs or simply directed flows. Directed diffusion, which uses directed flows, has thus been an effective tool in modelling traffic [8]. Directed diffusion was later successfully used in a transformer architecture, the Traffic Transformer [1].

Distance-based constructions of traffic networks do not however capture the long-distance, structural interactions of traffic networks [9]. Recent research on traffic prediction has thus explored deep-learning techniques to learn graph structures from the data itself [2, 12, 13, 17], an approach termed *graph estimation* [3]. Some graph estimation approaches augment the distance-based traffic graph with a learned structure [2, 4, 12], while others wholly learn the graph structure [13].

In this paper we propose a graph estimation strategy that uses a novel combination of the historical information supplied to the Traffic Transformer [1] to estimate separately a long-term and short-term structure, which we identify with the static and dynamic structures. It makes use of an attention mechanism to reduce the training parameters. Additionally we present a directed attention-based convolutional layer that models neighbourhoods as two separate groups of neighbours. The graph attention network (GAT) [14] does work on directed graphs, but it does not model neighbours with different orientations distinctly. We evaluate the proposed techniques on a modified Traffic Transformer against the METR-LA and PeMS-Bay datasets.

2 RELATED WORK

The task of traffic prediction means modelling temporal data in an irregular domain. A notable early example of the application of deep learning to this task is the diffusion convolutional recurrent neural network [8]. The temporal information is modelled using a recurrent neural network (RNN) and directed diffusion, where the in- and out-flows of a vertex are modelled as parallel diffusion processes. Alternatively the convolution has been defined such that it models spatial and temporal information simultaneously, a so-called gated convolutional neural network [16]. The Traffic Transformer [1] uses the directed diffusion layer in a transformer architecture that obviates the serial computation of RNNs, reporting state-of-the-art results. All three aforementioned works construct the traffic graph from sensor distances.

*Corresponding author.

Techniques to estimate the traffic graph have received greater attention in the literature recently. Generally speaking, there are three kinds of graph estimation strategies: metric-based methods, neural-based (hence indirect) methods and direct methods [3]. Metric-based methods use heuristics or some measure on the data to compute edge-weights, e.g. the cosine similarities of vertex attributes [6]. Neural-based techniques avoid relying exclusively on observed data by training a neural mechanism to estimate a graph from input. One model uses a separate graph-estimation component to compute a graph structure to complement an *a priori* structure [15]. More simply graph can be directly estimated that complements a pre-defined graph of the traffic network [4].

Alternatively the historical traffic information can be leveraged to learn different graphs, the structures of which complement one another; for example, by using two data pipelines to estimate static and dynamic traffic structures separately [7], or two levels of structure, macro- and micro-level, might complement one another [12]. The learning can be further decomposed into global and local structural estimation, each with static and dynamic components [17]. Typically that means the static component is fixed after training, while the dynamic component changes according to the sample fed to the model. The decomposition can also happen at a data-level, where a separate graph is learned on several streams of information, e.g. weekly, daily and hourly stream [2]. A full estimation of the n-by-n weight matrix can be avoided by using a global attention mechanism, an example of a neural-based approach [13]. Estimation is performed over multiple attention heads and the resulting structure is an average of the heads. In this example [13] there is decomposition into neither global/local nor static/dynamic structures.

3 METHODOLOGY

In this section we present *static–dynamic fusion*, an attention-based approach to graph estimation, and the *directed attention-based convolution*. Static–dynamic fusion uses two combinations of cyclic information, long-term and short-term, to estimate a graph, which is then used to structure the directed attention-based convolution. The local attention mechanism in the latter thus complements the global attention mechanism in the former.

Graph definitions. A *graph* $G = \langle V, E\rangle$ consists of a set of vertices V and edges E. The order of the graph is $n = |V|$ and the size is $m = |E|$. If two vertices $x, y \in V$ have an edge $xy \in E$, they are *adjacent*. E is an unordered set; when it is ordered, the edges acquire an *orientation* and the edges and the graph itself are *directed*. For a directed edge $\vec{xy} \in E$ x is the start vertex and y the endvertex.

For a given vertex $x \in V$, its adjacent vertices of undirected edges constitute the vertex's neighbourhood $\Gamma(x)$. When the graph is directed, the neighbourhood consists of the in-neighbourhood $\Gamma_{\text{in}}(x)$ for adjacencies where x is the start vertex and the out-neighbourhood Γ_{out} for adjacencies where y is the endvertex. By convention in deep learning $x \in \Gamma(x)$, but $x \notin \Gamma_{\text{in}}(x) \cup \Gamma_{\text{out}}(x)$.

If the vertices V are indexed, a graph can be represented by an adjacency matrix $\mathbf{A} \in \{0,1\}^{c\times c}$, where $\mathbf{A}_{ij}$ is non-zero when the i and jth vertices are adjacent. If the edges are weighted, where $\mathbf{A}$ is non-zero, $\mathbf{W}$ records a weight for each edge. Vertices may also carry c-dimensional signals, a mapping $f : V \longrightarrow \mathbb{R}^c$.

Static–dynamic fusion. The purpose of static–dynamic fusion is to estimate the graph structure from the data even at inference, so that the model does not rely on a fixed structure learned at training. We also want the model to draw on two sources of information: long-term, more stable structures, and short-term, ephemeral structures that occur locally. As with the Traffic Transformer [1], the data supplied to the graph estimation for prediction is separated into weekly-, daily- and hourly-periodic data, the tensors $f_w, f_d, f_h \in \mathbb{R}^{t\times n\times c}$, one week, one day and one hour preceding the target sequence respectively, where the data has t per sequence and c channels. The model learns to predict the target sequence Y by generating a predicted sequence $\hat{Y}$. The graph estimator learns a static structure $\mathbf{W}_{\text{stat}}$ and a dynamic structure $\mathbf{W}_{\text{dyn}}$ from two distinct compositions of the streams.

Before the composition, the streams are first projected into a new space by a projection matrix $\mathbf{U} \in \mathbb{R}^{c\times e}$ where the new space has e channels. Hence each vertex has a e-dimensional feature vector. The projected data is denoted $f_{w'} = f_w\mathbf{U}$, $f'_d = f_d\mathbf{U}$, $f'_h = f_h\mathbf{U}$.

The streams and target data are then concatenated along their first dimensions to yield the long- $\mathbf{Z}_l$ and short-term $\mathbf{Z}_s$ data, $\mathbf{Z}_l = [f'_w \;\|_0\; f'_d \;\|_0\; f'_h]$, and $\mathbf{Z}_s = [f'_h \;\|_0\; \zeta \cdot Y]$, where the binary operator $\|_i$ concatenates two tensors along their ith dimension, and $\zeta \in \{0,1\}$ is a flag indicating whether the algorithm is training, whether Y is part of the short-term data.

Two h-headed attention vectors, $\mathbf{a}_{\text{stat}}, \mathbf{a}_{\text{dyn}} \in \mathbb{R}^{2e\times h}$ are then applied to each pair of vertex feature vectors in the long- and short-term data to yield the two raw weight matrices, averaged over the timestep dimension:

$$\mathbf{W}'_{\text{stat},xy} = \sum_{i=0}^{3(t-1)} \left[\mathbf{Z}_{l,i,x} \,\middle\|\, \mathbf{Z}_{l,i,y}\right] \mathbf{a}_{\text{stat}}\,, \text{ and} \tag{1}$$

$$\mathbf{W}'_{\text{dyn},xy} = \sum_{i=0}^{(1+\zeta)(t-1)} \left[\mathbf{Z}_{s,i,x} \,\middle\|\, \mathbf{Z}_{s,i,y}\right] \mathbf{a}_{\text{dyn}}\,, \tag{2}$$

where $x, y \in V$. Finally the two weight matrices are combined with a sigmoid-activated, learned coefficient $\beta \in (0,1)$, one for each attention head, giving the estimated graph $\tilde{\mathbf{W}}$:

$$\tilde{\mathbf{W}} = \beta \cdot \text{ReLU}(\mathbf{W}_{\text{stat}}) + (1-\beta)\cdot \text{ReLU}(\mathbf{W}_{\text{dyn}}). \tag{3}$$

In order to sparsify the weight matrix, we impose a hard threshold of 0.1 on the values of $\tilde{\mathbf{W}}$. To restrain the size of the values we add L_1 regularisation of the sum of $\tilde{\mathbf{W}}$ to the loss function. Lastly we apply dropout to remove 80% of edges at training.

For the sake of the evaluation, we use a *simple graph estimation* strategy where there is no distinction between long- and short-term data, in order to determine the effectiveness of the static–dynamic fusion. In the simple graph estimation, there is no fusion as in Eq. (3). Instead, the weight matrix is estimated from a combination of all sequences using a single attention vector $\mathbf{a} \in \mathbb{R}^{2e\times h}$:

$$\tilde{\mathbf{W}}_{xy} = \text{ReLU}\left(\sum\nolimits_{i=0}^{(3+\zeta)(t-1)} \left[\mathbf{Z}_{i,x} \,\middle\|\, \mathbf{Z}_{i,y}\right] \mathbf{a}\right) \tag{4}$$

$$\text{where } \mathbf{Z} = \sum\nolimits_{i=0}^{t} \left[f'_w \,\|_0\, f'_d \,\|_0\, f'_h \,\|_0\, \zeta \cdot Y.\right], \tag{5}$$

Directed attention-based convolution. The attention-based convolution in the GAT computes coefficients that are applied over the sum of features in a given vertex x's neighbourhood $\Gamma(x)$. It does not distinguish between in- and out-flows, however. An input matrix of vertex signals $\mathbf{h} = f(G)$ is first projected with its own projection matrix $\mathbf{U} \in \mathbb{R}^{c \times e}$ to $\mathbf{h}' = \mathbf{hU} \in \mathbb{R}^{n \times e}$. The coefficients are then computed for the projected features using the attention vector $\mathbf{a} \in \mathbb{R}^{2e}$ and passed through a LeakyReLU function:

$$\mathbf{C}_{xy} = \text{LeakyReLU}([\mathbf{h}_x || \mathbf{h}_y]\mathbf{a}) \tag{6}$$

yielding a coefficient matrix $\mathbf{C}$. The coefficients are then normalised using softmax,

$$\alpha_{xy} = \text{softmax}_{\Gamma(x)} = \exp(\mathbf{C}_{xy}) / \sum\nolimits_{y \in \Gamma(x)} \exp \mathbf{C}_{xz}. \tag{7}$$

We want to learn the in- and out-flows separately, however, like the directed diffusion used in the Traffic Transformer [8]. We propose a directed attention-based convolution that leverages the power of attention and also models the in- and out-flows distinctly, as in the directed diffusion. Instead we increase the size of the dimension vector $\mathbf{a} \in \mathbb{R}^{3e}$ that is sliced into a vector for in-neighbours $\mathbf{a}_0$, the focal vertex $\mathbf{a}_1$ and out-neighbours $\mathbf{a}_2$. Then we reformulate Eq. (6) as

$$\mathbf{C}_{xy} = \begin{cases} \text{LeakyReLU}(c_x(x)) & x = y \\ \text{LeakyReLU}(c_x(\vec{yx}) + c_x(\vec{xy})) & x \neq y \end{cases} \tag{8}$$

$$\text{where, } c_x(x) = \mathbf{h}'_x \mathbf{a} \tag{9}$$

$$c_x(\vec{yx}) = \begin{cases} \left[\mathbf{h}'_y \middle\| \mathbf{h}'_x\right] \mathbf{a}_{0:1} & \vec{yx} \in E \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$c_x(\vec{xy}) = \begin{cases} \left[\mathbf{h}'_x \middle\| \mathbf{h}'_y\right] \mathbf{a}_{1:2} & \vec{xy} \in E \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where $a_{0:1}, a_{1:2} \in \mathbb{R}^{2e}$ are equal slices of $\mathbf{a}$. The simplicity of this mechanism adds a little overhead but allows the flows to be modelled separately. For our application, however, we include the edge weights learned in the graph estimation in the weighted sum for both the undirected and directed attention-based convolution. The weighted sum is thus: $h''_x = \text{LeakyReLU}(\sum_{y \in \Gamma(x)} \tilde{\mathbf{W}}_{xy} \alpha_{xy} h_y)$. When there is no graph learning, we still use the weights from the original graph $\mathbf{W}$. We refer to the edge weights from graph estimation as *global weights* and the attention coefficients in the convolutional layer as *local* weights.

4 EXPERIMENTAL DESIGN

Models. We evaluate the proposed techniques using the Traffic Transformer [1]. The model is fed either (1) the original distance-based graph or an estimated graph from (2) from static–dynamic fusion or (3) simple graph estimation. The Traffic Transformer's convolutional layers additionally are replaced with either (1) 1-step and (2) 2-step directed diffusion, (3) undirected attention or (4) directed attention. In total, we evaluate the performance of 12 models. We treat the 2-step directed diffusion with no graph estimation as our baseline because such a configuration is essentially the Traffic Transformer as originally presented [1]. We refer to the models using undirected or directed attention, structured with the estimated graphs, as the "fully attention-based models".

Control variables. The models were trained on 4 NVIDIA V100 GPUs and 16 Intel Xeon Gold 6148 CPU cores hosted by the Sunbird supercomputer at Swansea University. Each model was trained for 100 epochs with a batch-size of 4 for the METR-LA dataset and 2 for the PeMS-Bay dataset. Batch normalisation was applied to the output of the convolutional layer. The batch size must be kept low because the graph estimator demands a large memory capacity owing to so many parameters. We used the Adam optimiser with a learning rate $\lambda = 1 \times 10^{-4}$. For the scheduled sampling, we choose a sampling rate that means that at the fifth epoch there is a fifty-fifty chance that the model trains its decoder on the true values or on predictions. As the loss function we used the mean absolute error (MAE), and add a regularisation term of the estimated graph $\tilde{\mathbf{W}}$:

$$||\tilde{\mathbf{W}}|| = \frac{1}{b} \sum\nolimits_{i=1}^{b} \sum\nolimits_{j=1}^{n} \sum\nolimits_{k=1}^{n} |\tilde{\mathbf{W}} i, j, k|, \tag{12}$$

yielding the loss function: $l(y, \hat{y}) = \text{MAE}(y, \hat{y}) + ||\tilde{\mathbf{W}}||$. The input data to the encoder and decoder is z-normalised, where the mean and standard deviation are computed from the training set. The labels are not normalised; therefore the model is trained to map normalised speeds to unnormalised speeds.

Datasets and metrics. We used the METR-LA and PeMS-Bay datasets. The METR-LA dataset is a network of 207 loop detectors in Los Angeles County. The dataset contains 23,974, 3,425 and 6,850 training, validation and testing sequences respectively. The PeMS-Bay dataset consists of speed readings 325 loop detectors from the San Francisco Bay Area [8]. The dataset is split into 36,465, 5,209 and 10,419 training, validation and testing sequences respectively. The samples of both datasets are sequentially ordered but randomly sampled to train the model. Each successive sample's prediction horizon overlaps by 55 minutes with the previous sample's.

The graphs that are published with METR-LA and PeMS-Bay are computed using thresholded Gaussian kernel [10] using the distances between each pair of vertices $\text{dist}(a, b)$:

$$\mathbf{W}_{xy} = \begin{cases} \exp\left(-\text{dist}(a, b)^2 / \sigma^2\right) & \text{if dist}(a, b) \leq \kappa, \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

where σ is the standard deviation of the training samples, κ is a threshold and $\mathbf{W}_{xy}$ is the entry at the xth row and yth column. This computation leads to some disconnected vertices. We remove these vertices before we conduct our experiments, yielding reduced graphs that we still refer to as the original graphs. We also remove self-loops. The reduced graph of METR-LA has 206 instead of 207 vertices and that of PeMS-Bay has 319 instead of 325 vertices.

The models are assessed on the basis of three metrics, lower values of all representing a better prediction. The first metric is the MAE, which measures the average absolute deviation of the predicted value from the target value. The second metric, the root mean squared error, measures the averaged squared prediction errors and the quality of the predictions, namely how far from the target value the predicted values lie. The third metric, the mean average percentage error (MAPE), the ratio of the prediction error with respect to the target value. We follow the example of Cai et al. [1] by excluding datapoints in the measurement where the true value $y(i, j)$ is zero for two reasons: (1) no prediction is necessary at these points; and (2) a zero speed would lead to a zero-division error in computing the MAPE.

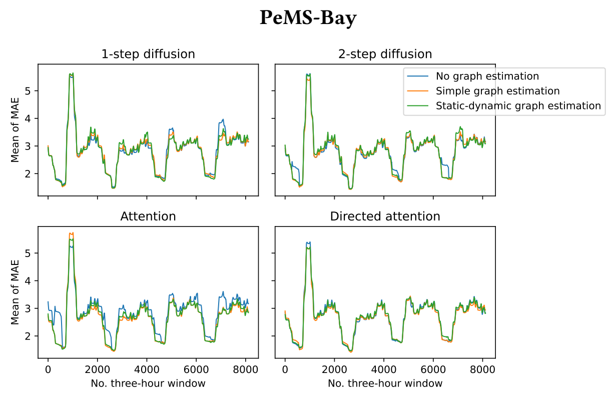

Figure 1: A linegraph plotting the rolling average in three-hour windows on the test set of the PeMS-Bay dataset. In a few places graph estimation flattens spikes in the error.

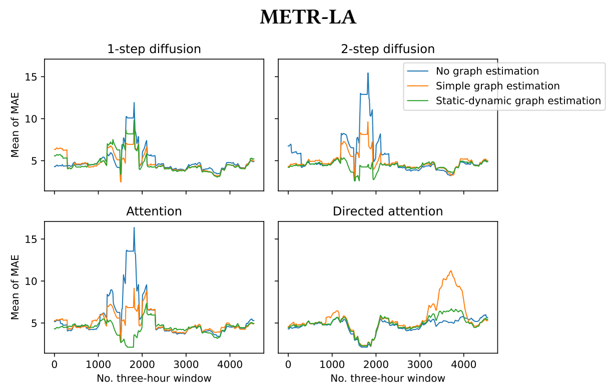

Figure 2: A linegraph plotting the rolling average in three-hour windows on the test set of the METR-LA dataset. Note that between the 1,000th and 3,000th hours the error consistently rises except in the directed attention model.

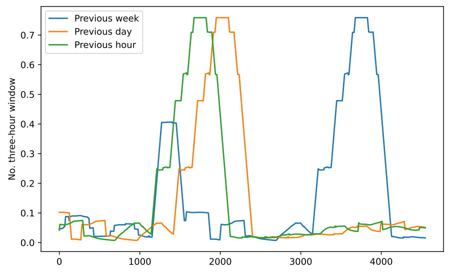

Figure 3: The proportion of sensors with missing data was computed at each timestep and a three-hour rolling average was taken over the sequence.

5 RESULTS AND DISCUSSION

The full tables of results on the test set of the PeMS-Bay and METR-LA datasets are presented respectively in Table 1. We have also visualised the MAE as a rolling average of three-hour windows for the PeMS-Bay and METR-LA datasets respectively in Figs. 1 and 2.

We can see from the results on the PeMS-Bay dataset that directed attention attains the lowest average where there is no graph estimation. Directed diffusion is a runner-up. Static–dynamic fusion appears to worsen results, whereas the simple graph estimation improves the performance. The two-week errors suggest that both simple graph estimation and static–dynamic fusion are correcting errors where they occur in other models. They are minor corrections, but they are consistent.

The results on the METR-LA dataset indicate however that directed diffusion is not attaining the lowest average errors. Moreover, static–dynamic fusion is apparently ameliorating the error. Interestingly, 1-step directed diffusion attains the lowest average errors when static-dynamic fusion is used. A more complex picture emerges in the two-week errors. Although it does not have the lowest errors, the directed diffusion is correcting for errors that consistently occur in the other models when predicting between windows 1000 and 3000. On analysing the proportion of vertices without data, it is clear that in periods where there is a high quantity of missing data, the models without graph estimation suffer Fig. 3. This suggests that graph estimation, in particular static–dynamic fusion, is robust to missing data. Additionally the static–dynamic fusion is correcting the other models in periods of missing data. It does not substantially improve the directed diffusion model in this same timespan, with or without graph estimation. The downside is that there is unusual some difficulty around the 300th window, which static–dynamic fusion better corrects for, but directed attention better corrects for.

It is worth noting in the discussion of these results that the METR-LA dataset is considered more difficult than PeMS-Bay [8], which is noticeable in the models' higher prediction errors on the former and the noted difficulties in prediction visualised in Fig. 2. Nevertheless a few provisional conclusions may be made. It is clear that directed diffusion is more robust to errors and that static–dynamic fusion is rendering the other models more robust to errors, too. A reason for the robustness may lie in the 80% dropout used during training. If the graph estimator is drawing upon many redundant structures to make traffic predictions, it would avoid the flaw of drawing on one road where unexpected conditions occur. Although, if this is the case, it would not satisfactorily explain the difference between the directed diffusion without graph estimation in comparison to the other convolutional layers.

The method is limited in a few ways however. A study of the estimated graphs themselves is beyond the scope of this paper, but a cursory inspection shows us that the estimated graphs are dense in comparison to the topological graphs. Our proposed means to limit the weights of the graph did not work. This had consequences for the efficiency of graph estimation alongside traffic prediction. In future we will conduct a fuller study of the graphs. There is also potential here to simplify the whole model by implicitly including the global attention coefficients from graph estimation in the convolutional layer, instead of constructing a weight matrix which is then used as usual.

6 CONCLUSION

In this paper we presented a novel graph estimation strategy built on novel combinations of historical sequence data at two different levels. One layer represented long-term changes in the graph

Table 1: The prediction errors of each model on PeMS-Bay (above) and METR-LA (below) at 15, 30 and 60 minutes. The best results for each dataset appear in bold, the second-best in italics. The model indicated by the asterisk (*) is our baseline in both datasets (Traffic Transfomrer [1]).

Models on PeMS-Bay		MAE			MAPE			RMSE		
		15'	30'	60'	15'	30'	60'	15'	30'	60'
No graph estimation	Attention	2.787	2.841	2.885	6.575	6.686	6.773	4.810	4.915	5.018
	Directed attention	**2.652**	**2.681**	**2.704**	6.450	**6.505**	**6.533**	**4.656**	**4.719**	**4.782**
	1-step diffusion	2.730	2.791	2.837	**6.424**	6.573	6.684	4.711	4.827	4.931
	2-step diffusion*	2.741	2.782	2.812	6.452	6.537	6.594	4.757	4.844	4.924
Static-dynamic graph estimation	Attention	**2.634**	**2.660**	**2.680**	**6.368**	**6.406**	**6.430**	**4.640**	**4.693**	**4.751**
	Directed attention	2.659	2.680	2.710	6.476	6.479	6.510	4.647	4.699	4.777
	1-step diffusion	2.759	2.798	2.835	6.518	6.594	6.663	4.769	4.853	4.943
	2-step diffusion	2.743	2.782	2.812	6.565	6.656	6.723	4.759	4.840	4.920
Simple graph estimation	Attention	**2.615**	**2.642**	**2.659**	6.386	6.424	**6.438**	4.631	4.682	**4.732**
	Directed attention	2.627	2.657	2.684	**6.331**	**6.383**	6.439	**4.598**	**4.668**	4.743
	1-step diffusion	2.754	2.802	2.836	6.522	6.611	6.673	4.770	4.870	4.961
	2-step diffusion	2.718	2.756	2.784	6.410	6.485	6.536	4.717	4.800	4.879

Models on METR-LA		MAE			MAPE			RMSE		
		15'	30'	60'	15'	30'	60'	15'	30'	60'
No graph estimation	Attention	4.894	5.089	5.463	13.457	13.877	14.603	7.515	7.798	8.285
	Directed attention	4.880	4.717	**4.695**	13.926	13.541	**13.476**	7.755	7.678	**7.748**
	1-step diffusion	**4.657**	**4.686**	4.927	13.521	**13.486**	13.852	**7.415**	**7.495**	7.812
	2-step diffusion*	5.008	5.160	5.415	**13.318**	13.541	13.937	7.653	7.893	8.253
Static-dynamic graph estimation	Attention	4.308	**4.314**	4.438	12.452	12.328	12.492	7.119	7.137	7.326
	Directed attention	4.980	4.869	4.914	14.350	13.915	13.907	7.846	7.836	8.016
	1-step diffusion	**4.289**	4.456	4.771	**11.892**	**12.173**	12.784	**6.852**	7.097	7.531
	2-step diffusion	4.374	4.336	**4.413**	12.503	12.251	**12.251**	7.133	**7.083**	**7.208**
Simple graph estimation	Attention	4.670	4.719	5.007	12.748	12.756	13.214	7.326	7.460	7.842
	Directed attention	5.521	5.403	5.520	14.766	14.351	14.488	8.356	8.366	8.604
	1-step diffusion	**4.230**	**4.353**	**4.732**	**11.656**	**11.880**	**12.577**	**6.766**	**6.989**	**7.473**
	2-step diffusion	4.529	4.632	4.924	13.629	13.781	14.229	7.363	7.440	7.760

and the other short-term changes. At the core of the graph estimation strategy is an attention mechanism, which serves to reduce the number of trainable parameters in the model. The attention mechanism in the graph estimator is a global estimation of graph structure. It is complemented by the second contribution of this paper, a directed attention-based convolution, where the attention coefficients are local. The two levels complement one another, and in the results it is shown that they are robust to errors to which the other models are consistently vulnerable. It is suggested that the dropout layer in the graph estimation enables the estimator to detect redundant structures. In future work we will investigate strategies to enforce a greater deal of sparsity. We will also investigate means of incorporating the global attention coefficients into the convolutional layers, obviating the needless construction of whole weight matrices.

REFERENCES

[1] Ling Cai, Krzysztof Janowicz, Gengchen Mai, Bo Yan, and Rui Zhu. 2020. Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. *Transactions in GIS* 24, 3 (2020), 736–755.

[2] Liujuan Chen, Kai Han, Qiao Yin, and Zongmai Cao. 2020. GDCRN: Global Diffusion Convolutional Residual Network for Traffic Flow Prediction. In *Knowledge Science, Engineering and Management (Lecture Notes in Computer Science).* 438–449.

[3] Stavros Georgousis, Michael P. Kenning, and Xianghua Xie. 2021. Graph Deep Learning: State of the Art and Challenges. *IEEE Access* 9 (2021), 22106–22140.

[4] Kan Guo, Yongli Hu, Zhen Qian, Yanfeng Sun, Junbin Gao, and Baocai Yin. 2022. Dynamic Graph Convolution Network for Traffic Forecasting Based on Latent Network of Laplace Matrix Estimation. *IEEE Transactions on Intelligent Transportation Systems* 23, 2 (2022), 1009–1018.

[5] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems.* 1024–1034.

[6] Ali Golzadeh Kermani, Ali Kamandi, and Ali Moeini. 2022. Integrating graph structure information and node attributes to predict protein-protein interactions. *Journal of Computational Science* 64 (2022), 101837.

[7] Xiangyuan Kong, Weiwei Xing, Xiang Wei, Peng Bao, Jian Zhang, and Wei Lu. 2020. STGAT: Spatial-Temporal Graph Attention Networks for Traffic Flow Forecasting. *IEEE Access* 8 (2020), 134363–134372.

[8] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations.* 1–16.

[9] Zhishuai Li, Gang Xiong, Yuanyuan Chen, Yisheng Lv, Bin Hu, Fenghua Zhu, and Fei-Yue Wang. 2019. A Hybrid Deep Learning Approach with GCN and LSTM for Traffic Flow Prediction*. In *IEEE Intelligent Transportation Systems Conference.* 1929–1933.

[10] David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* (2013).

[11] Tengfei Song, Wenming Zheng, Peng Song, and Zhen Cui. 2019. EEG Emotion Recognition Using Dynamical Graph Convolutional Neural Networks. *IEEE Transactions on Affective Computing* (2019), 1–1.

[12] Xuxiang Ta, Zihan Liu, Xiao Hu, Le Yu, Leilei Sun, and Bowen Du. 2022. Adaptive Spatio-temporal Graph Neural Network for traffic forecasting. *Knowledge-Based Systems* (2022), 1–34.

[13] Cong Tang, Jingru Sun, Yichuang Sun, Mu Peng, and Nianfei Gan. 2020. A General Traffic Flow Prediction Approach Based on Spatial-Temporal Graph Attention. *IEEE Access* 8 (2020), 153731–153741.

[14] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations.* 1–12.

[15] Ruijia Wang, Shuai Mou, Xiao Wang, Wanpeng Xiao, Qi Ju, Chuan Shi, and Xing Xie. 2021. Graph Structure Estimation Neural Networks. In *Web Conference 2021.* 342–353.

[16] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI International Joint Conference on Artificial Intelligence.* 3634–3640.

[17] Qi Zhang, Jianlong Chang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2020. Spatio-Temporal Graph Structure Learning for Traffic Forecasting. *AAAI Conference on Artificial Intelligence* 34, 01 (2020), 1177–1185.