FC-GAGA: Fully Connected Gated Graph Architecture for Spatio-Temporal Traffic Forecasting

Boris N. Oreshkin,[1] Arezou Amini, [2] Lucy Coyle [2]
Mark J. Coates [2]
[1]Unity Technologies, [2]McGill University
boris.oreshkin@gmail.com, arezou.amini@mail.mcgill.ca, lucy.coyle@mail.mcgill.ca, mark.coates@mcgill.ca

Abstract

Forecasting of multivariate time-series is an important problem that has applications in traffic management, cellular network configuration, and quantitative finance. A special case of the problem arises when there is a graph available that captures the relationships between the time-series. In this paper we propose a novel learning architecture that achieves performance competitive with or better than the best existing algorithms, without requiring knowledge of the graph. The key element of our proposed architecture is the learnable fully connected hard graph gating mechanism that enables the use of the state-of-the-art and highly computationally efficient fully connected time-series forecasting architecture in traffic forecasting applications. Experimental results for two public traffic network datasets illustrate the value of our approach, and ablation studies confirm the importance of each element of the architecture. The code is available here: https://github.com/boreshkinai/fc-gaga.

1 Introduction

Many multivariate time-series (TS) forecasting problems naturally admit a graphical model formulation. This is especially true when the entities whose past is observed and whose future has to be predicted affect each other through simple causal relationships. For example, introducing pepsi products in a store will very likely decrease future sales of coca-cola; car traffic congestion at one point on a highway is likely to slow down the traffic at preceding highway segments. Without graphical modeling, the model is blind to these nuances, making entity interactions a collection of confounding factors, extremely hard for the model to explain and predict. Equipped with a learnable model for entity properties (e.g. entity embeddings), a model for entity interactions (e.g. graph edge weights), and a mechanism to connect them to a TS model (e.g. a gating mechanism), we can learn the otherwise unknown entity interactions to improve forecasting accuracy.

Problems amenable to graphical TS modeling include forecasting demand for related products (Singh et al. 2019), electricity demand (Rolnick et al. 2019), road traffic (Shi et al. 2020) or passenger demand (Bai et al. 2019). Recent studies have shown that models that explicitly account for the underlying relationships across multiple TS outperform models that forecast each TS in isolation. Although the inclusion of graph modeling has proven to improve accuracy, current models have several serious limitations. First, the complexity and therefore runtime of these models is significantly higher. Second, some models rely on the definition of relationships between variables provided by a domain expert (*e.g.* an adjacency matrix is heuristically defined based on the geographical relationships between observed variables). Finally, existing models tend to rely on Markovian assumptions to make modelling the interactions across variables tractable.

To address these limitations we propose a novel architecture, called FC-GAGA, that is based on a combination of a fully-connected TS model N-BEATS (Oreshkin et al. 2020) and a hard graph gate mechanism proposed in this paper. To produce the forecast for a single TS (node in the graphical model), it weighs the historical observations of all other nodes by learnable graph weights, gates them via a ReLU and then stacks gated observations of all nodes to process them via fully connected residual blocks (see Fig. 1). The advantages of this architecture are threefold. First, the architecture does not rely on the knowledge of the underlying graph focusing on learning all the required non-linear predictive relationships instead. Second, the basic layer of the architecture is stackable and we allow every layer to learn its own graph structure. This endows the model with the ability to learn a very general non-Markovian information diffusion process that can be learned effectively, which we show empirically. Finally, FC-GAGA is a very memory and computation efficient architecture, which we demonstrate via profiling. Ablation studies demonstrate that when using the efficient fully-connected residual time-series prediction module, it is not sufficient to use standard graph attention — the sparsification achieved by our proposed novel graph gate is essential in achieving good predictive performance.

1.1 Problem Statement

Let a graph $G = (V,E)$ be defined as an ordered collection of vertices, $V = 1,\dots,N$, and edges, $E \subset V \times V$. We are interested in the multivariate TS forecasting problem defined on this graph. Each vertex v in the graph is assumed to generate a sequence of observations, $\mathbf{y}_v = [y_{v,1}\dots,y_{v,T}] \in \mathbb{R}^T$, governed by an unknown stochastic random process. The graph connectivity encoded in E is assumed to capture unknown relations between the vertices. For example, the graph

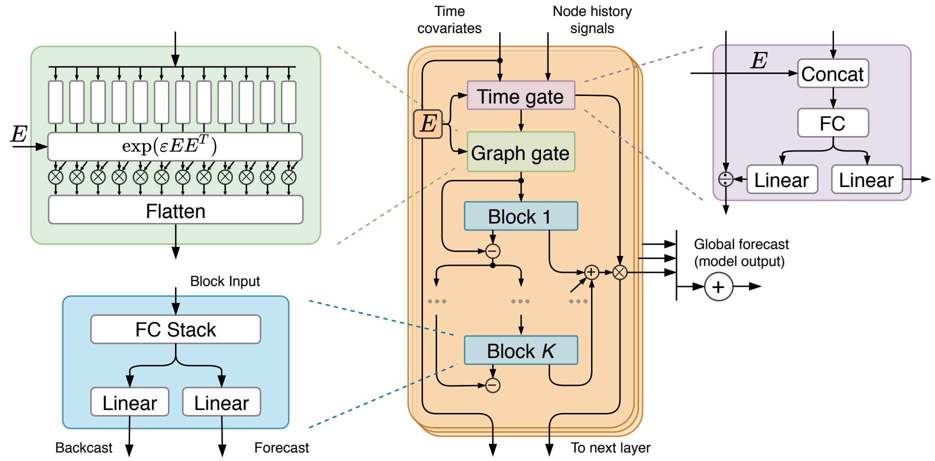

Figure 1: FC-GAGA block diagram each layer includes a graph gate, a time gate and a fully connected time-series model.

edges E may reflect the connectivity of roads in a road network and $\mathbf{y}_v$ may be the sequence of observations of traffic velocity. The task is to predict the vector of future values $\mathbf{y}_v \in \mathbb{R}^H = [y_{T+1}, y_{T+2}, \dots, y_{T+H}]$ for every vertex v based on the observations generated by all the vertices in the graph up to time T. The model input of length $w \leq T$ at vertex v, ending with the last observed value $y_{v,T}$, is denoted $\mathbf{x}_v \in \mathbb{R}^w = [y_{v,T-w+1}, \dots, y_{v,T}]$. We denote $\widehat{\mathbf{y}}_v$ the point forecast of $\mathbf{y}_v$ at vertex v.

Metrics: We measure accuracy via Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE): $\mathrm{MAE} = \frac{1}{N}\sum_{v=1}^{N} |y_{v,T+H} - \widehat{y}_{v,T+H}|$, $\mathrm{MAPE} = \frac{1}{N}\sum_{v=1}^{N} \frac{|y_{v,T+H} - \widehat{y}_{v,T+H}|}{|y_{v,T+H}|}$, and $\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{v=1}^{N}(y_{v,T+H} - \widehat{y}_{v,T+H})^2}$.

1.2 Summary of Contributions

We propose a novel principle of combining a fully connected state-of-the-art univariate TS forecasting model N-BEATS (Oreshkin et al. 2020) with a learnable time gate and a learnable hard graph gate mechanisms. We empirically show that the proposed model learns the graph parameters effectively from the data and achieves impressive predictive performance. We show that the proposed model offers computational advantage and reduces the training time by at least a factor of three relative to models with similar accuracy.

2 FC-GAGA

The block diagram of the proposed architecture is presented in Fig. 1. We model node i by representing it as an embedding vector of dimensionality d, $\mathbf{e}_i = [e_{i,1}, \dots e_{i,d}]$. The collection of all such vectors comprises node embedding matrix $\mathbf{E} \in \mathbb{R}^{N \times d}$. In the following, we describe the operation of a single layer, dropping the layer index for clarity.

Graph edge weights The strengths of node links are encoded in a weight matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ derived from node embeddings:

$$\mathbf{W} = \exp(\varepsilon \mathbf{E}\mathbf{E}^T). \tag{1}$$

Here ε is a parameter that is set to allow for the decoupling of the scaling of $\mathbf{E}$, which is used in other parts of the architecture, from the scaling that is required to achieve the necessary dynamic range in $\mathbf{W}$. We expect that the magnitudes of edge weights $\mathbf{W}_{i,j}$ will reflect the strength of mutual influence between the pair of nodes (i, j) at a given FC-GAGA layer.

Time gate block The time gate block models the time covariate features (*e.g.* time-of-day, day-of-week, etc.) that may be available together with the node observations. We propose to model time related features using a multiplicative gate model that divides/multiplies the input/output of the FC-GAGA layer by time effects derived from the time feature via a fully connected network as depicted in Fig. 1. Additionally, the input time feature vector is concatenated with the node embedding to account for the fact that each node may have a different seasonality pattern. This is equivalent to removing a node-specific multiplicative seasonality from the input of the block and applying it again at the output of the block. We allow the input and output time effects to be decoupled via separate linear projection layers, because in general time at the input and at the output is different.

Graph gate block The input to the FC-GAGA layer is a matrix $\mathbf{X} \in \mathbb{R}^{N \times w}$ containing the history of length w of all nodes in the graph. We denote by $\widetilde{\mathbf{x}}$ the maximum of the input values over the time dimension, $\widetilde{\mathbf{x}}_i = \max_j \mathbf{X}_{i,j}$. The gating operation produces matrix $\mathbf{G} \in \mathbb{R}^{N \times Nw}$. Row i of the gated matrix corresponds to node i and it contains all the information accumulated by the graph during past w steps:

$$\mathbf{G}_{i,j+k} = \mathrm{RELU}[(\mathbf{W}_{i,j}\mathbf{X}_{j,k} - \widetilde{\mathbf{x}}_i)/\widetilde{\mathbf{x}}_i]. \tag{2}$$

Graph gate relates the information collected by nodes i, j via two mechanisms. First, the measurements in nodes i and j are related to each other by subtraction and levelling operations inside RELU. Furthermore, the RELU operation has the function of shutting off the irrelevant i, j pairs while not affecting the scale alignment achieved via $\mathbf{W}_{i,j}$. The magnitude of $\mathbf{W}_{i,j}$ affects the probability of opening the hard gate. Our empirical study shows that the magnitude of $\mathbf{W}_{i,j}$ correlates well with the spatial proximity of nodes (see Figures 3 and 4). We found that without the hard gating the graph weighting is not effective. We believe this is due to the fact that for each target node there are only a few nodes that are relevant at a given layer, so the input to the fully connected architecture is supposed to be sparse. Hard gating encourages sparsity. Soft gating provides input that is not sparse, overwhelming a fully connected network with too many low-magnitude inputs originating from many nodes in the graph. Additionally, according to the complexity analysis presented at the end of this section, our graph gate design has N times smaller complexity, $O(N^2)$, compared to the approaches known in the literature that are based on matrix multiplication in the graph diffusion step $O(N^3)$ (e.g. DCRNN and Graph WaveNet).

Fully connected time-series block We propose a fully connected residual architecture with L hidden layers, R residual blocks and weights shared across nodes. Its input for node i, $\mathbf{Z}_i$, is conditioned on the node embedding and its own history: $\mathbf{Z} = [\mathbf{E}, \mathbf{X}/\widetilde{\mathbf{x}}, \mathbf{G}]^T$, $\mathbf{Z} \in \mathbb{R}^{N(w+1)+d \times N}$. Using residual block and layer superscripts and denoting the fully connected layer with weights $\mathbf{A}^{r,\ell}$ and biases $\mathbf{b}^{r,\ell}$ as $\mathrm{FC}_{r,\ell}(\mathbf{H}^{r,\ell-1}) \equiv \mathrm{ReLU}(\mathbf{A}^{r,\ell}\mathbf{H}^{r,\ell-1} + \mathbf{b}^{r,\ell})$, the operation of the fully connected residual TS modeling architecture is described as follows:

$$\begin{aligned}
\mathbf{Z}^r &= \mathrm{ReLU}[\mathbf{Z}^{r-1} - \widehat{\mathbf{Z}}^{r-1}], \\
\mathbf{H}^{r,1} &= \mathrm{FC}_{r,1}(\mathbf{Z}^r),\ \ldots,\ \mathbf{H}^{r,L} = \mathrm{FC}_{r,L}(\mathbf{H}^{r,L-1}), \\
\widehat{\mathbf{Z}}^r &= \mathbf{B}^r\mathbf{H}^{r,L},\ \widehat{\mathbf{Y}}^r = (\mathbf{H}^{r,L})^T\mathbf{F}^r.
\end{aligned} \tag{3}$$

We assume $\widehat{\mathbf{Z}}^0 \equiv \mathbf{0}$, $\mathbf{Z}^0 \equiv \mathbf{Z}$; projection matrices have dimensions $\mathbf{B}^r \in \mathbb{R}^{N(w+1)+d \times d_h}$, $\mathbf{F}^r \in \mathbb{R}^{d_h \times H}$ and the final forecast is the sum of forecasts of all residual blocks, $\widehat{\mathbf{Y}} = \sum_r \widehat{\mathbf{Y}}^r$.

FC-GAGA layer stacking is based on the following three principles. First, the next layer accepts the sum of forecasts of previous layers as input. Second, each FC-GAGA layer has its own set of node embeddings and thus its own graph gate. Thus each layer is provided a freedom to gate the information flow across nodes in accordance with the processing already accomplished by the previous layer. For example, in the first FC-GAGA layer, for node id 5, it may be optimal to focus on the histories of node ids [10, 200, 500]. However, since the first FC-GAGA layer updates the states of all nodes, node 5 may no longer need the information provided by nodes [10, 200, 500], nor by their neighbours; and instead may wish to focus on node ids [3 and 15], as they now provide more important information. This is clearly a more flexible information diffusion model than the Markov model based on node proximity that is common in the traffic forecasting literature (Li et al. 2018). Finally, the final model output is equal to the average of layer forecasts.

Complexity analysis In the following analysis we skip the batch dimension and compute the complexity involved in creating a single forecast of length H for all nodes N in the graph when the input history is of length w, the node embedding width is d and the hidden layer width is d_h. Analysis details can be found in Appendix B. The graph gate block has complexity $O(N^2(w+d))$, as is evident from eq. (2). The time gate mechanism producing a seasonality factor for each node using its associated time feature scales linearly with the number of nodes, the hidden dimension, the input history length: $O(N(d+w)d_h)$. Finally, the fully-connected TS model with L FC layers and R residual blocks that accepts the flattened input $N \times Nw$ has complexity $O(R(2N^2wd_h + (L-2)Nd_h^2))$. In most practical configurations, the total complexity of the model will be dominated by $O(N^2Rwd_h)$.

3 Empirical Results

Datasets FC-GAGA is evaluated on two traffic datasets, METR-LA and PEMS-BAY (Chen et al. 2001; Li et al. 2018) consisting of the traffic speed readings collected from loop detectors and aggregated over 5 minute intervals. METR-LA contains 34,272 time steps of 207 sensors collected in Los Angeles County over 4 months. PEMS-BAY contains 52,116 time steps of 325 sensors collected in the Bay Area over 6 months. The datasets are split in 70% training, 10% validation, and 20% test, as defined in (Li et al. 2018).

Baselines We compare FC-GAGA both with temporal models that do not require a pre-specified graph and spatio-temporal models that may rely on a pre-specified graph or have a learnable graph. The following univariate temporal models provided by (Li et al. 2018) are considered: ARIMA (Makridakis and Hibon 1997), implemented using a Kalman filter; SVR (Wu, Ho, and Lee 2004), a linear Support Vector Regression model; FNN, a Feedforward Neural Network; and FC-LSTM (Sutskever, Vinyals, and Le 2014), a sequence-to-sequence model that uses fully connected LSTMs in encoder and decoder. The spatio-temporal models include DCRNN (Li et al. 2018) (Diffusion Convolutional Recurrent Neural Network, a graph convolutional network inside the sequence-to-sequence architecture); STGCN (Yu, Yin, and Zhu 2018) (Spatio-Temporal Graph Convolutional Network, merges graph convolutions with gated temporal convolutions); Graph WaveNet (Wu et al. 2019), fuses graph convolution and dilated causal convolution; GMAN (Zheng et al. 2020) (Graph Multi-Attention Network, an encoder-decoder model with multiple spatio-temporal attention blocks, and a transform attention layer between the encoder and the decoder); STGRAT (Park et al. 2019) (Spatio-Temporal Graph Attention Network for Traffic Forecasting, an encoder-decoder model using the positional encoding method of the Transformer (Vaswani et al. 2017) to capture features of long sequences and node attention to capture spatial correlation) . Of these methods, only Graph Wavenet can generate predictions without a pre-specified graph. For DCRNN, we report the results after bug fix in the code, which are better than the reported results in the paper. For STGCN, Graph WaveNet, GMAN, and STGRAT we use the settings and report results from the original papers.

FC-GAGA architecture details and training setup Scalar ε in (1) is set to 10. The embedding dimensionality, d, is set to 64 and the hidden layer width d_h for all fully connected layers is set to 128. The number of layers L in the fully-connected TS model is equal to 3 and the number of blocks R is equal to 2. We use weight decay of 1e-5 to regularize fully-connected layers. The model is trained using

Dataset	Models	15 min			30 min			60 min		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
METR-LA	DCRNN	2.67	6.84%	5.17	3.08	8.38%	6.30	3.56	10.30%	7.52
	STGCN	2.88	7.62%	5.74	3.47	9.57%	7.24	4.59	12.70%	9.40
	Graph WaveNet	2.69	6.90%	5.15	3.07	8.37%	6.22	3.53	10.01%	7.37
	GMAN	2.77	7.25%	5.48	3.07	8.35%	6.34	3.40	9.72%	7.21
	STGRAT	2.60	6.61%	5.07	3.01	8.15%	6.21	3.49	10.01%	7.42
	ARIMA	3.99	9.60%	8.21	5.15	12.70%	10.45	6.90	17.40%	13.23
	SVR	3.99	9.30%	8.45	5.05	12.10%	10.87	6.72	16.70%	13.76
	FNN	3.99	9.90%	7.94	4.23	12.90%	8.17	4.49	14.00%	8.69
	FC-LSTM	3.44	9.60%	6.30	3.77	10.90%	7.23	4.37	13.20%	8.69
	Graph WaveNet‡	2.80	7.45%	5.45	3.18	9.00%	6.42	3.57	10.47%	7.29
	FC-GAGA	2.75	7.25%	5.34	3.10	8.57%	6.30	3.51	10.14%	7.31
	FC-GAGA(4 layers)‡	**2.70**	**7.01%**	**5.24**	**3.04**	**8.31%**	**6.19**	**3.45**	**9.88%**	**7.19**
PEMS-BAY	DCRNN	1.31	2.74%	2.76	1.66	3.76%	3.78	1.98	4.74%	4.62
	STGCN	1.36	2.90%	2.96	1.81	4.17%	4.27	2.49	5.79%	5.69
	Graph WaveNet	1.30	2.73%	2.74	1.63	3.67%	3.70	1.95	4.63%	4.52
	GMAN	1.34	2.81%	2.82	1.62	3.63%	3.72	1.86	4.31%	4.32
	STGRAT	1.29	2.67%	2.71	1.61	3.63%	3.69	1.95	4.64%	4.54
	ARIMA	1.62	3.50%	3.30	2.33	5.40%	4.76	3.38	8.30%	6.50
	SVR	1.85	3.80%	3.59	2.48	5.50%	5.18	3.28	8.00%	7.08
	FNN	2.20	5.19%	4.42	2.30	5.43%	4.63	2.46	5.89%	4.98
	FC-LSTM	2.05	4.80%	4.19	2.20	5.20%	4.55	2.37	5.70%	4.96
	Graph WaveNet‡	**1.34**	**2.79%**	2.83	1.69	3.79%	3.80	2.00	4.73%	4.54
	FC-GAGA	1.36	2.87%	2.86	1.68	3.80%	3.80	1.97	4.67%	4.52
	FC-GAGA(4 layers)‡	**1.34**	2.82%	**2.82**	**1.66**	**3.71%**	**3.75**	**1.93**	**4.48%**	**4.40**

Table 1: Error metrics computed using the standard protocol (Wu et al. 2019) (average over last time step of horizon, input window length 12). Lower numbers are better. ‡Graph Wavenet trained using official code by the authors using only adaptive matrix without the support of geographical adjacency matrix. ‡ FC-GAGA(4 layers) is the proposed model with 4 layers. In the fourth layer, the graph gate weights are set to the identity matrix, implying more reliance on the pure time-series component.

the Adam optimizer with default tensorflow settings and initial learning rate of 0.001 for 60 epochs. The learning rate is annealed by a factor of 2 every 6 epochs starting at epoch 43. One epoch consists of 800 batches of size 4 and the model takes the history of 12 points and predicts 12 points (60 min) ahead in one shot. Each training batch is assembled using 4 time points chosen uniformly at random from the training set and the histories of all nodes collected at each of the time points. METR-LA has 207 sensor nodes and in PEMS-BAY has 325, resulting in the batches consisting of $207 \cdot 4 = 828$ and $325 \cdot 4 = 1300$ time-series, respectively. The objective function used to train the network is MAE, averaged over all nodes and all forecasts within horizon $H = 12$:

$$\mathcal{L} = \frac{1}{HN} \sum_{i=1}^{H} \sum_{v=1}^{N} |y_{v,T+i} - \widehat{y}_{v,T+i}|. \quad (4)$$

Quantitative results Our key empirical results appear in Table 1. FC-GAGA compares favourably even against graph-based models that rely on additional external graph definitions on both METR-LA and PEMS-BAY datasets (DCRNN, STGCN, Graph WaveNet, and GMAN). Most of the time, FC-GAGA outperforms Graph WaveNet model when they are trained and evaluated in the same conditions, i.e. both models only rely on the graph learned from the data (Graph WaveNet is using only the *adaptive* adjacency matrix that it learns from the data). It significantly outperforms the univariate models (ARIMA, SVR, FNN, and FC-LSTM). Note that STGRAT heavily relies on the main ingredients of Transformer architecture such as positional encoding and attention mechanisms. Therefore, comparing FC-GAGA against it gives a good idea of how our approach stands against Transformer-based methods in terms of accuracy.

Qualitative results The final FC-GAGA forecast is composed of the average of the forecasts of individual layers. Figure 2 shows the contributions of different layers to the final 15 min ahead forecast (after scaling by the averaging factor $1/3$). We can see that the role of the first layer is mostly to provide a baseline forecast, while at the same time accounting for some seasonal effects. The layer 2 contribution to the prediction clearly captures daily seasonality. Layer 2 and especially layer 3 provide iterative correction terms to the original baseline produced by layer 1, based on the most recent data. This is especially evident for layer 3 whose output is inactive most of the time, becoming active when significant correction is required because the observed signals undergo significant stochastic changes in short periods of time.

Next, we show in Fig. 3 the geographical distribution of the weights $\mathbf{W}_{i,j}$ in the graph gate, specified in eq. (2), for layers 1–3, as learned for the METR-LA dataset. Each layer is provided the freedom to learn its own relationship across graph

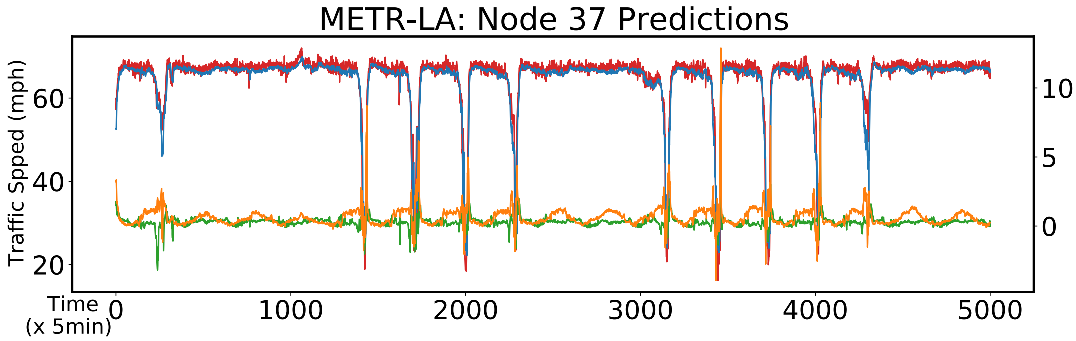

Figure 2: FC-GAGA 15 min ahead forecasts for node 37 in METR-LA dataset. Blue, green and orange lines depict the partial forecasts produced by layers 1, 2, and 3 of the architecture respectively. Blue & red: left axis; orange & green: right axis. Additional results appear in Appendix A.

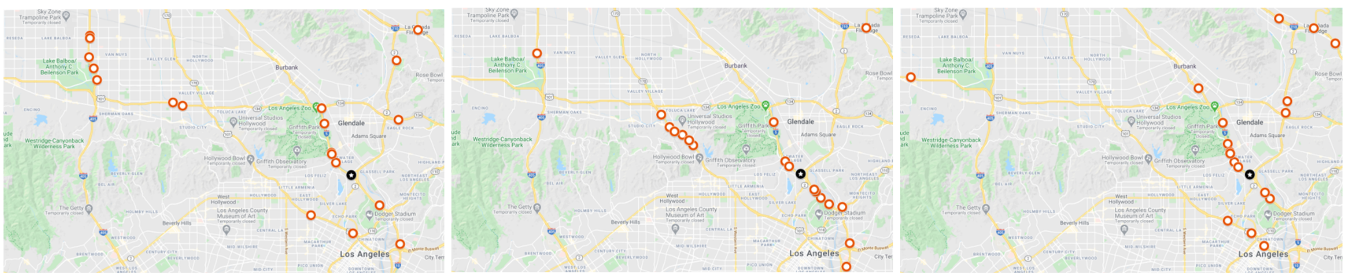

Figure 3: Maps of 20 highest weighted nodes for layers 1, 2, and 3. Black star is the forecasted node.

nodes; the learned relationships differ significantly across layers, indicating information aggregation from different spatial regions. In Fig. 4 (left) we observe that the gating is less strictly enforced in the first layer (the average $\mathbf{W}_{i,j}$ values are higher in the first layer) and the geographic distribution of values is more dispersed (see Fig. 3, left). We interpret this as indicating that in layer 1 FC-GAGA collects information across a wide variety of nodes and geographical locations to construct a stable baseline forecast. As we move from layer 2 to layer 3, we can see that the nodes with highest graph weights more tightly concentrate around the target node for which the forecast is produced (see Fig. 3, middle and right and Fig. 4, right). Fig. 4 (left) indicates that many more $\mathbf{W}_{i,j}$ have smaller values progressively in layers 2 and 3, implying stricter gating in eq. (2). Our interpretation of this is that to provide iterative updates to the baseline forecast, FC-GAGA focuses on the nodes that are closer to the target node and restricts the information flow such that the correction terms are defined by the nodes with the most relevant information.

Ablation studies Our ablation studies validate the effectiveness of the FC-GAGA layer stacking mechanism, the graph gating mechanism and the time gate. Table 2 demonstrates the performance of FC-GAGA as a function of the number of layers. Increasing the number of layers leads to substantial improvement on the METR-LA dataset, while on PEMS-BAY the number of layers does not affect performance significantly. METR-LA is known to be a harder problem than PEMS-BAY because of the more erratic nature of its TS. This implies that increasing the number of FC-GAGA layers to solve harder problems may bring additional accuracy benefits while using only one FC-GAGA layer to solve an easier problem may be benefitial from the computational efficiency standpoint (the runtime scales approximately linearly with the number of layers). The final row in the table ($4\mathrm{I}^{\ddagger}$) shows the performance when the fourth layer is set to the identity, so that the layer focuses on forming a prediction using only the history of each node. This approach leads to a noticeable improvement; forcing one layer to learn univariate relationships can be beneficial.

The top section of Table 3 shows the results of ablating the graph gate and time gate mechanisms with a 3-layer FC-GAGA network. Both the time gate and graph gate individually lead to improvements over a straightforward multivariate N-BEATS model and then combine to offer further improvement. The bottom section of the table examines different approaches for the graph gate. “graph attention” is a standard graph attention approach that does not perform hard gating. We see that the sparsification provided by our proposed gate is essential; graph attention is even outperformed by the univariate FC-GAGA model (“identity”). The univariate FC-GAGA outperforms all univariate methods in Table 1 by a large margin. When $\mathbf{W}$ is set to all ones (“ones”), FC-GAGA can learn relationships between different nodes, but it cannot emphasize influential nodes. We examine three learnable options: “shared learnable” where all layers share a single learnable $\mathbf{W}$, “learnable first layer” where $\mathbf{W}$ associated with

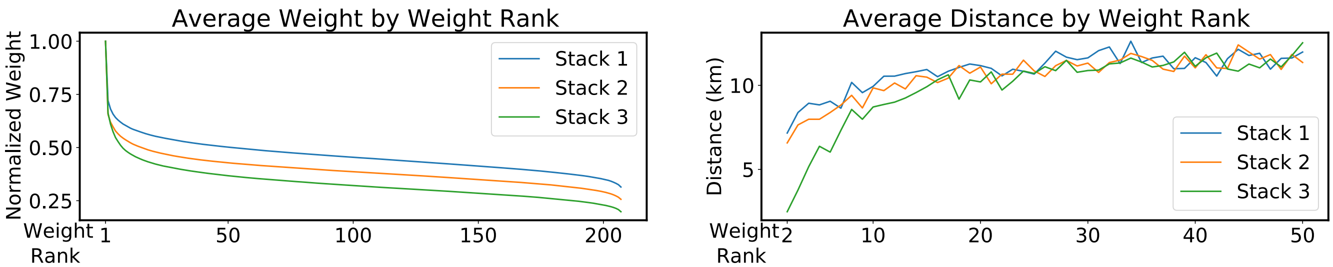

Figure 4: Average of graph gate weights $\mathbf{W}_{i,j}$ normalized by the self-weight $\mathbf{W}_{i,i}$ (left) and their average distances from the forecasted node (right) per FC-GAGA layer in METR-LA dataset.

		15 min			30 min			60 min		
Dataset	Layers	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
METR-LA	1	2.80	7.36%	5.44	3.17	8.82%	6.44	3.63	10.55%	7.48
	2	2.77	7.30%	5.37	3.13	8.74%	6.39	3.54	10.41%	7.36
	3	2.75	7.25%	5.34	3.10	8.57%	6.30	3.51	10.14%	7.31
	4	2.75	7.21%	5.34	3.10	8.54%	6.34	3.52	10.19%	7.34
	4I‡	2.70	7.01%	5.24	3.04	8.31%	6.19	3.45	9.88%	7.19
PEMS-BAY	1	1.35	2.85%	2.85	1.69	3.85%	3.83	2.00	4.78%	4.61
	2	1.36	2.87%	2.86	1.68	3.80%	3.81	1.97	4.64%	4.52
	3	1.36	2.87%	2.86	1.68	3.80%	3.80	1.97	4.67%	4.52
	4	1.35	2.83%	2.86	1.69	3.78%	3.83	1.98	4.66%	4.57
	4I‡	1.34	2.82%	2.82	1.66	3.71%	3.75	1.93	4.48%	4.40

Table 2: Ablation study: the effectiveness of the FC-GAGA layer stacking. 4I‡ uses the identity for the fourth layer weight matrix.

the first layer is learnable and it is set to the ones matrix for other layers, and the fully learnable FC-GAGA approach. Allowing the architecture to learn a different weight matrix for each layer leads to the best prediction performance, and the additional computational overhead is very minor.

Profiling results To confirm FC-GAGA's computational efficiency we conducted a profiling experiment using a P100 GPU in the default Google Colab environment. We profiled the original codes provided by the authors of DCRNN (Li et al. 2018) and Graph Wavenet (Wu et al. 2019). We profiled our tensorflow 2.0 implementation of FC-GAGA, which relies on standard Keras layer definitions, with no attempt to optimize for memory or speed. Table 4 clearly shows that FC-GAGA is more computationally effective as it consumes approximately half the memory and compute time of Graph WaveNet and is about 10 times faster than DCRNN and about 5-10 times more memory efficient. We can also see that it scales well between METR-LA (207 nodes) and PEMS-BAY (325 nodes) datasets, which may be an important property for handling larger scale problems with thousands of nodes.

4 Related Work

Multivariate TS prediction or forecasting has been studied intensively for decades. Historically, neural network approaches struggled to compete with state-of-the-art statistical forecasting models. Recently, several neural network architectures that are trained on many time series, but then form predictions for a single variable based on its past history (and covariates) have eclipsed statistical methods (Salinas et al. 2019; Oreshkin et al. 2020; Smyl 2020). In contrast to our work, these architectures do not simultaneously form forecasts for multiple time series using past information from all of them. Other methods use multiple input time-series to predict a single target TS (Bao, Yue, and Rao 2017; Qin et al. 2017; Lai et al. 2018; Guo and Lin 2018; Chang et al. 2018). For these architectures, several innovations have proven effective, including attention mechanisms to determine which input variables and time lags to focus on (Qin et al. 2017; Guo and Lin 2018; Munkhdalai et al. 2019; Liu, Lu, and Cai 2020). In this vein, the transformer architecture is modified in (Li et al. 2019) to address TS forecasting and DeepGLO (Sen, Yu, and Dhillon 2019) is a hybrid model that combines regularized matrix factorization to derive factors with a temporal convolution network for local prediction.

Graph-based models In some settings, we are provided with a graph that is thought to capture the relationships between the variables. The neural network architectures usually combine graph convolutional networks (GCNs), which can focus on spatial relationships, with GRUs, LSTMs, TCNs, or RNNs (Zhao et al. 2019; Li et al. 2018; Huang et al. 2019; Yu, Yin, and Zhu 2018; Chen et al. 2019). A few approaches apply graph-based learning directly to a spatio-temporal graph (Yu et al. 2019; Song et al. 2020). Performance can be improved using attention mechanisms (Guo et al. 2019; Bai et al. 2019; Park et al. 2019; Zheng et al. 2020; Shi et al. 2020). More advanced architectures also offer an avenue for improvement. ST-UNet (Yu, Yin, and Zhu 2019) employs spatio-temporal pooling/unpooling to allow the architecture to learn represen-

Dataset	Models	Layers	15 min MAE	15 min MAPE	15 min RMSE	30 min MAE	30 min MAPE	30 min RMSE	60 min MAE	60 min MAPE	60 min RMSE
M-LA	(1) MV N-BEATS	3	3.00	7.96%	5.90	3.60	10.22%	7.28	4.44	13.59%	8.92
	(2) add time gate	3	2.86	7.60%	5.61	3.24	9.13%	6.66	3.68	10.81%	7.67
	(3) add graph gate	3	2.81	7.33%	5.36	3.21	8.75%	6.36	3.67	10.51%	7.45
	FC-GAGA	3	2.75	7.25%	5.34	3.10	8.57%	6.30	3.51	10.14%	7.31
PB	(1) MV N-BEATS	3	1.41	2.94%	3.05	1.86	4.20%	4.26	2.40	5.90%	5.48
	(2) add time gate	3	1.37	2.87%	2.89	1.70	3.83%	3.86	1.99	4.68%	4.57
	(3) add graph gate	3	1.35	2.84%	2.86	1.69	3.79%	3.82	2.00	4.72%	4.58
	FC-GAGA	3	1.36	2.87%	2.86	1.68	3.80%	3.80	1.97	4.67%	4.52
	Graph gate **W**										
M-LA	graph attention	3	2.99	7.90%	5.83	3.56	10.00%	7.15	4.43	13.15%	8.89
	identity	3	2.97	7.80%	5.87	3.54	9.88%	7.25	4.35	12.77%	8.93
	ones	3	2.87	7.71%	5.67	3.24	9.22%	6.71	3.67	10.80%	7.65
	shared learnable	3	2.77	7.20%	5.36	3.13	8.53%	6.35	3.57	10.09%	7.37
	learnable first layer	3	2.77	7.28%	5.40	3.13	8.67%	6.41	3.55	10.23%	7.44
	FC-GAGA	3	2.75	7.25%	5.34	3.10	8.57%	6.30	3.51	10.14%	7.31
PB	graph attention	3	1.44	3.00%	3.08	1.92	4.32%	4.38	2.57	6.07%	5.86
	identity	3	1.41	2.92%	3.05	1.86	4.10%	4.27	2.44	5.58%	5.66
	ones	3	1.38	2.89%	2.89	1.70	3.82%	3.86	2.00	4.70%	4.60
	shared learnable	3	1.37	2.95%	2.88	1.72	4.00%	3.90	2.01	4.84%	4.62
	learnable first layer	3	1.36	2.87%	2.86	1.69	3.82%	3.83	1.99	4.70%	4.57
	FC-GAGA	3	1.36	2.87%	2.86	1.68	3.80%	3.80	1.97	4.67%	4.52

Table 3: Ablation study: the effectiveness of the FC-GAGA graph gate and time gate

	METR-LA	
	Runtime, min	GPU memory, GB
DCRNN	358	8.63
Graph WaveNet	90	2.14
FC-GAGA, 3 layers	37	0.93
	PEMS-BAY	
	Runtime, min	GPU memory, GB
DCRNN	828	8.63
Graph WaveNet	192	2.75
FC-GAGA, 3 layers	69	1.47

Table 4: Profiling results: total training and evaluation runtime, and GPU memory utilization. Measured using official DCRNN and Graph WaveNet codes and our non-optimized tensorflow implementation of FC-GAGA on NVIDIA P100 GPU in the default Google Colab environment.

tations at multiple scales. Graph WaveNet (Wu et al. 2019) employs dilated causal convolution to extract temporal relationships from a larger perceptive field. Xu et al. (Xu et al. 2020) introduce spatial-temporal transformer networks. The graphs are often derived from considerations that do not correspond exactly to the prediction task. For example, a graph for traffic forecasting might be based on geographical proximity. It can be beneficial to learn an appropriate graph from the data. The algorithms in (Yu et al. 2019; Diao et al. 2019) learn the graph as a pre-processing step. A better approach is to combine graph learning with prediction; the architectures in (Wu et al. 2019; Zhang et al. 2020) incorporate mechanisms to learn adaptive adjacency matrices that are subsequently used in GCNs. Although GraphWaveNet can learn an adjacency matrix via a graph attention approach, this is not a suitable approach when combined with the fully-connected time-series prediction module, as shown in Table 3. Our proposed graph gating mechanism has an important sparsification effect that prevents overfitting. The fully-connected architecture is attractive, because it has very good generalization and is much less demanding in terms of computation and memory.

5 Conclusions

We proposed and empirically validated a novel neural architecture for spatio-temporal forecasting, which we call FC-GAGA (Fully Connected Gated Graph Architecture). FC-GAGA combines a fully connected TS model with temporal and graph gating mechanisms, that are both generally applicable and computationally efficient. We empirically demonstrate that the proposed model can be learned efficiently from the data to capture non-Markovian relations across multiple variables over layers in the architecture, resulting in excellent generalization performance. We further profile FC-GAGA's training and inference runtime and demonstrate that it is several times more efficient in the utilization of GPU memory and compute than existing models with comparable accuracy. Our results provide compelling positive evidence to stimulate the development of fully connected architectures for graph based information processing.

References

Bai, L.; Yao, L.; Kanhere, S.; Wang, X.; and Sheng, Q. 2019. STG2Seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting. In *Proc. IJCAI*, 1981–1987.

Bao, W.; Yue, J.; and Rao, Y. 2017. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS One* 12(7).

Chang, Y.-Y.; Sun, F.-Y.; Wu, Y.-H.; and Lin, S.-D. 2018. A memory-network based solution for multivariate time-series forecasting. *arXiv preprint arXiv:1809.02105* .

Chen, C.; Li, K.; Teo, S. G.; Zou, X.; Wang, K.; Wang, J.; and Zeng, Z. 2019. Gated Residual Recurrent Graph Neural Networks for Traffic Prediction. In *Proc. AAAI*, 485–492.

Chen, C.; Petty, K.; Skabardonis, A.; Varaiya, P.; and Jia, Z. 2001. Freeway Performance Measurement System: Mining Loop Detector Data. *Transportation Research Record* 1748(1): 96–102.

Diao, Z.; Wang, G.; Zhang, D.; Liu, Y.; Xie, K.; and He, S. 2019. Dynamic Spatial-Temporal Graph Convolutional Neural Networks for Traffic Forecasting. In *Proc. AAAI*.

Guo, S.; Lin, Y.; Feng, N.; Song, C.; and Wan, H. 2019. Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting. In *Proc. AAAI*, volume 33, 922–929.

Guo, T.; and Lin, T. 2018. Multi-variable LSTM neural network for autoregressive exogenous model. *arXiv preprint arXiv:1806.06384* .

Huang, Y.; Weng, Y.; Yu, S.; and Chen, X. 2019. Diffusion Convolutional Recurrent Neural Network with Rank Influence Learning for Traffic Forecasting. In *Proc. IEEE Int. Conf. Big Data Science And Engineering*, 678–685.

Lai, G.; Chang, W.-C.; Yang, Y.; and Liu, H. 2018. Modeling long-and short-term temporal patterns with deep neural networks. In *Proc. ACM Int. Conf. Research & Development in Information Retrieval*, 95–104.

Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.-X.; and Yan, X. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems*, 5244–5254.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *Proc. Int. Conf. Learning Representations*.

Liu, F.; Lu, Y.; and Cai, M. 2020. A Hybrid Method With Adaptive Sub-Series Clustering and Attention-Based Stacked Residual LSTMs for Multivariate Time Series Forecasting. *IEEE Access* 8: 62423–62438.

Makridakis, S.; and Hibon, M. 1997. ARMA models and the Box–Jenkins methodology. *Journal of Forecasting* 16(3): 147–163.

Munkhdalai, L.; Munkhdalai, T.; Park, K. H.; Amarbayasgalan, T.; Erdenebaatar, E.; Park, H. W.; and Ryu, K. H. 2019. An end-to-end adaptive input selection with dynamic weights for forecasting multivariate time series. *IEEE Access* 7: 99099–99114.

Oreshkin, B. N.; Carpov, D.; Chapados, N.; and Bengio, Y. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *ICLR*.

Park, C.; Lee, C.; Bahng, H.; Won, T.; Kim, K.; Jin, S.; Ko, S.; and Choo, J. 2019. STGRAT: A Spatio-Temporal Graph Attention Network for Traffic Forecasting. *arXiv preprint arXiv:1911.13181* .

Qin, Y.; Song, D.; Chen, H.; Cheng, W.; Jiang, G.; and Cottrell, G. W. 2017. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. In *Proc. IJCAI*.

Rolnick, D.; Donti, P. L.; Kaack, L. H.; Kochanski, K.; Lacoste, A.; Sankaran, K.; Ross, A. S.; Milojevic-Dupont, N.; Jaques, N.; Waldman-Brown, A.; Luccioni, A.; Maharaj, T.; Sherwin, E. D.; Mukkavilli, S. K.; Körding, K. P.; Gomes, C.; Ng, A. Y.; Hassabis, D.; Platt, J. C.; Creutzig, F.; Chayes, J.; and Bengio, Y. 2019. Tackling Climate Change with Machine Learning. *CoRR* abs/1906.05433.

Salinas, D.; Flunkert, V.; Gasthaus, J.; and Januschowski, T. 2019. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *Int. J. Forecasting* .

Sen, R.; Yu, H.-F.; and Dhillon, I. S. 2019. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *Adv. Neural Information Processing Systems*, 4838–4847.

Shi, X.; Qi, H.; Shen, Y.; Wu, G.; and Yin, B. 2020. A spatial-temporal attention approach for traffic prediction. *IEEE Trans. Intelligent Transportation Systems* 1–10.

Singh, P. K.; Gupta, Y.; Jha, N.; and Rajan, A. 2019. Fashion Retail: Forecasting Demand for New Items. *CoRR* abs/1907.01960.

Smyl, S. 2020. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting* 36(1): 75 – 85.

Song, C.; Lin, Y.; Guo, S.; and Wan, H. 2020. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proc. AAAI*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Adv. Neural Information Processing Systems*, 3104–3112.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. In *Advances in Neural Information Processing Systems*, 5998–6008.

Wu, C.-H.; Ho, J.-M.; and Lee, D. T. 2004. Travel-time prediction with support vector regression. *IEEE Trans. Intelligent Transportation Systems* 5(4): 276–281.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *Proc. IJCAI*, 1907–1913.

Xu, M.; Dai, W.; Liu, C.; Gao, X.; Lin, W.; Qi, G.-J.; and Xiong, H. 2020. Spatial-Temporal Transformer Networks for Traffic Flow Forecasting. *arXiv preprint arXiv:2001.02908* .

Yu, B.; Li, M.; Zhang, J.; and Zhu, Z. 2019. 3d graph convolutional networks with temporal graphs: A spatial information free framework for traffic forecasting. *arXiv preprint arXiv:1903.00919* .

Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proc. IJCAI*.

Yu, B.; Yin, H.; and Zhu, Z. 2019. ST-UNet: A Spatio-Temporal U-Network for Graph-structured Time Series Modeling. *arXiv e-prints, arXiv:1903.05631* .

Zhang, Q.; Chang, J.; Meng, G.; Xiang, S.; and Pan, C. 2020. Spatio-Temporal Graph Structure Learning for Traffic Forecasting. In *Proc. AAAI*.

Zhao, L.; Song, Y.; Zhang, C.; Liu, Y.; Wang, P.; Lin, T.; Deng, M.; and Li, H. 2019. T-GCN: A temporal graph convolutional network for traffic prediction. *IEEE Trans. Intelligent Transportation Systems* .

Zheng, C.; Fan, X.; Wang, C.; and Qi, J. 2020. GMAN: A Graph Multi-Attention Network for Traffic Prediction. In *Proc. AAAI*.