MULTIVARIATE TIME SERIES FORECASTING WITH LATENT GRAPH INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper introduces a new architecture for multivariate time series forecasting that simultaneously infers and leverages relations among time series. We cast our method as a modular extension to univariate architectures where relations among individual time series are dynamically inferred in the latent space obtained after encoding the whole input signal. Our approach is flexible enough to scale gracefully according to the needs of the forecasting task under consideration. In its most straight-forward and general version, we infer a potentially fully connected graph to model the interactions between time series, which allows us to obtain competitive forecast accuracy compared with the state-of-the-art in graph neural networks for forecasting. In addition, whereas previous latent graph inference methods scale $O(N^2)$ w.r.t. the number of nodes N (representing the time series), we show how to configure our approach to cater for the scale of modern time series panels. By assuming the inferred graph to be bipartite where one partition consists of the original N nodes and we introduce K nodes (taking inspiration from low-rank-decompositions) we reduce the time complexity of our procedure to $O(NK)$. This allows us to leverage the dependency structure with a small trade-off in forecasting accuracy. We demonstrate the effectiveness of our method for a variety of datasets where it performs better or very competitively to previous methods under both the fully connected and bipartite assumptions.

1 INTRODUCTION

Time Series Forecasting (TSF) has been widely studied due to its practical significance in a wide variety of applications such as climate modelling (Mudelsee, 2019), supply chain management in retail (Larson, 2001; Böse et al., 2017), market analysis in finance (Andersen et al., 2005), traffic control (Li et al., 2017) and medicine (Kaushik et al., 2020). Petropoulos et al. (2020) provide a non-systematic overview of further applications. In TSF, given a sequence of data points indexed over time, we aim to estimate its future values based on previously observed data. Data is often multivariate, meaning that multiple variables vary over time, each variable may not only depend on its own historical values, but also on other variables' past. Efficiently modelling the dependencies between these variables is still an open problem.

Multivariate Time Series (MTS) methods aim to leverage the dependencies between variables in order to improve the forecasting accuracy. Some classical MTS forecasting algorithms such as Vector Autoregression (VAR) (Lütkepohl, 2005) or Gaussian Processes (Roberts et al., 2013) only consider linear dependencies among variables. A natural way to model non-linear dependencies in deep learning is via Graph Neural Networks (GNNs) (Bruna et al., 2013; Defferrard et al., 2016; Kipf and Welling, 2016). In fact, GNNs have been successfully applied for the Multivariate Time Series forecasting (Li et al., 2017; Yu et al., 2017; Chen et al., 2020a), leveraging the relations among different series. But these methods require a pre-defined adjacency matrix which may only be available in some specific datasets, for example, traffic datasets, where it can be constructed from the spatial structure of a city. More recently, a family of methods that do not require a pre-defined adjacency matrix have been proposed (Wu et al., 2019; Franceschi et al., 2019; Wu et al., 2020; Shang et al., 2021). In this case, a latent graph representation is inferred while forecasting, allowing to operate on a larger variety of datasets. Our work belongs to this category.

However, inferring all pairwise relations may come at a higher computational cost, as the number of relations scales quadratically $O(N^2)$ w.r.t. the number of nodes. This limits the scalability to large datasets. Additionally, the above mentioned latent graph inference methods perform message passing update at every time step iteration which can also be expensive.

To overcome these limitations, we propose a new latent graph inference algorithm for Multivariate Time Series forecasting that is more efficient than previous algorithms while achieving better or competitive performance. We cast the latent graph inference as a modular and easy-to-implement extension to current univariate models. The graph is dynamically inferred per inputted data stream allowing a more flexible representation than a static graph for the whole dataset. Additionally, we optionally reduce the complexity from $O(N^2)$ (Fully Connected Assumption) to $O(NK)$ (Bipartite Assumption) where $K \ll N$ for a small trade off in performance.

2 BACKGROUND

2.1 TIME SERIES FORECASTING

In time series forecasting we want to estimate a future time series $\mathbf{x}_{t+1:T}$ given its past $\mathbf{x}_{t_0:t}$ and (optionally) some context information $\mathbf{c}$ where $t_0 \leq t \leq T$ indexes over time. For the multivariate case we assume the time series is composed of N variates at a time such that $\mathbf{x}_{t_0:T} = \{\mathbf{x}_{1,t_0:T}, \ldots, \mathbf{x}_{N,t_0:T}\} \in \mathbb{R}^{N \times T - t_0 + 1}$. In this section we distinguish two main categories of time series forecasting methods, Global Univariate and Multivariate.

Global Univariate methods: In this case we only use the past of each univariate to predict its future. However, the model weights θ_u are shared across all univariate time series. More formally:

$$\hat{\mathbf{x}}_{i,t+1:T} = f_u(\mathbf{x}_{i,t_0:t}, \mathbf{c}_i; \theta_u) \tag{1}$$

where $\hat{\mathbf{x}}$ denotes the estimated values, $i \in \{1, \ldots, N\}$ indexes over multivariates and $f_u(\cdot)$ is the estimator function with learnable parameters θ_u shared across time series. Conditioning on the past of each univariate may limit the performance of the forecasting algorithm compared to multivariate ones. Despite that, it simplifies the design of f_θ and already provides reasonable results. A popular example of univariate models would be (Salinas et al., 2020).

Multivariate methods: Multivariate methods condition on all past data (all N variates) and directly predict the multivariate target. More formally:

$$\hat{\mathbf{x}}_{t+1:T} = f_m(\mathbf{x}_{t_0:t}, \mathbf{c}). \tag{2}$$

Different variables may be correlated and/or depend on the same con-founders. For example, in retail forecasting, PPE masks and antibacterial soaps jointly increased in demand during the early days of the COVID-19 pandemic. In traffic forecasting, an increase of the outcome traffic flow in a given neighborhood may result in an increase of the income traffic flow on another one. Modelling these dependencies may improve the forecasting accuracy, but it may come at a cost of higher complexity and hence more expensive algorithms, specially when trying to model all pairwise interactions between variates.

2.2 GRAPH NEURAL NETWORKS

Graph Neural Networks (GNNs) (Bruna et al., 2013; Defferrard et al., 2016; Kipf and Welling, 2016) operate directly on graph structured data. They have gained a lot of attention in the last years due to their success in a large variety of domains which benefit from modelling interactions between different nodes/entities. In the context of multivariate time series, GNNs can be used to model the interactions between time series. In this work we consider the type of GNN introduced by (Gilmer et al., 2017). Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $v_i \in \mathcal{V}$ and edges $e_{ij} \in \mathcal{E}$, we define a graph convolutional layer as:

$$\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i^l, \mathbf{h}_j^l) \qquad \mathbf{m}_i = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{m}_{ij} \qquad \mathbf{h}_i^{l+1} = \phi_h(\mathbf{h}_i^l, \mathbf{m}_i) \tag{3}$$

Where ϕ_e and ϕ_h are the edge and node functions, usually approximated as Multi Layer Perceptrons (MLPs), $\mathbf{h}_i^l \in \mathbb{R}^{\text{nf}}$ is the nf-dimensional embedding of a node v_i at layer l and $\mathbf{m}_{ij}$ is the edge

embedding that propagates information from node v_j to v_i. A GNN is constructed by stacking multiple of these Graph Convolutional Layers. Additionally, in Equation 3 we include $\alpha_{ij} \in (0, 1)$ which is a scalar value that performs the edge inference over the neighbors similarly to Veličković et al. (2017). As done in (Satorras et al., 2021), we choose this value to be computed as the output of a function $\alpha_{ij} = \phi_\alpha(\mathbf{m}_{ij})$ where ϕ_α is composed of just a linear layer followed by at sigmoid activation function.

3 Related Work

Time series forecasting has been extensively studied in the past due to its practical significance with a number of recent overview articles available (et al., 2020; Benidis et al., 2020; Lim and Zohren, 2021). Traditionally, most classical methods are univariate in nature (see e.g., Hyndman and Athanasopoulos (2017) for an overview). While some of these have multi-variate extensions (e.g., ARMA and VARMA models), they are limited by the amount of related time series information they can incorporate. Dynamic factor models (Geweke, 1977; Wang et al., 2019a) are fore-runners of a family of models that has recently received more attention, the so-called global models (Januschowski et al., 2019; Montero-Manso and Hyndman, 2022). These global models estimate their parameters over an entire panel of time series, so thereby taking advantage of cross time series learning, but still produce a univariate forecast. Many such global models have been proposed building on the main neural network architectures like RNNs (Salinas et al., 2020; Liberty et al., 2020; Bandara et al., 2019), CNNs (Wen et al., 2017; Chen et al., 2020b), Transformers (Li et al., 2019; Lim et al., 2021; Eisenach et al., 2020) and also combining classical probabilistic models with deep learning (Rangapuram et al., 2018; Kurle et al., 2020; de Bézenac et al., 2020). However, these global models do not explicitly model the relationship between the time series in the panel.

Most recently, global multi-variate forecasting models have received attention, in particular models that attempt to capture the relationship of the time series via a multi-variate likelihood (Rasul et al., 2020; 2021; de Bézenac et al., 2020; Salinas et al., 2019). Here, we attempt to capture the multi-variate nature of many modern forecasting problems primarily by using a multi-variate time series as input. For this, a natural way to model and exploit the relationship between time series is via Graph Neural Networks (Bruna et al., 2013; Defferrard et al., 2016; Kipf and Welling, 2016) which have been successfully applied to a wide variety of deep learning domains where exploiting relations between entities/nodes can benefit the prediction task. Even in those cases where edges are not explicitly provided in the dataset, attention or a latent graph can be inferred from the node embeddings such that GNNs can still leverage the structure of the data. Some examples of latent graph inference or attention are (Wang et al., 2019b) in point clouds, (Franceschi et al., 2019) in semi-supervised graph classification, (Kipf et al., 2018) in modelling dynamical systems, (Kazi et al., 2020) in zero-shot learning and 3D point cloud segmentation,(Garcia and Bruna, 2017; Kossen et al., 2021) in image classification, (Cranmer et al., 2020) in inferring symbolic representations and (Fuchs et al., 2020; Satorras et al., 2021) in molecular property prediction.

In Multivariate Time Series (MTS) forecasting we can leverage dependencies between time series by exchanging information among them. Lai et al. (2018); Shih et al. (2019) are some of the first deep learning approaches designed to exploit those pair-wise dependencies. More recent methods, (Li et al., 2017; Yu et al., 2017; Seo et al., 2018; Zhao et al., 2019) are built in the intersection of Graph Neural Networks and time series forecasting but they require a pre-defined adjacency matrix. Lately, new methods that infer a latent graph from the node embeddings have been introduced in MTS forecasting (Wu et al., 2020; Shang et al., 2021), and in MTS anomaly detection (Zhang et al., 2020; Deng and Hooi, 2021). These methods can be applied to any dataset even when there is not an explicitly defined adjacency matrix. But this comes with a limitation, inferring the edges can be expensive since those scale quadratically $O(N^2)$ w.r.t the number of nodes/variables N.

Our approach is related to (Wu et al., 2020; Shang et al., 2021), but in contrast a) Our latent graph is dynamically inferred for each inputted data stream instead of a static graph for the whole dataset which allows a more flexible graph representation. b) It is modular since it can be added as an extension to standard univariate methods after encoding the input signal $\mathbf{x}_{t_0:t}$, this also makes the graph operation cheaper since the message exchange is only done at time step t. c) Optionally reduces the number of edges from $O(N^2)$ to $O(NK)$ where $K \ll N$ with the bipartite assumption.

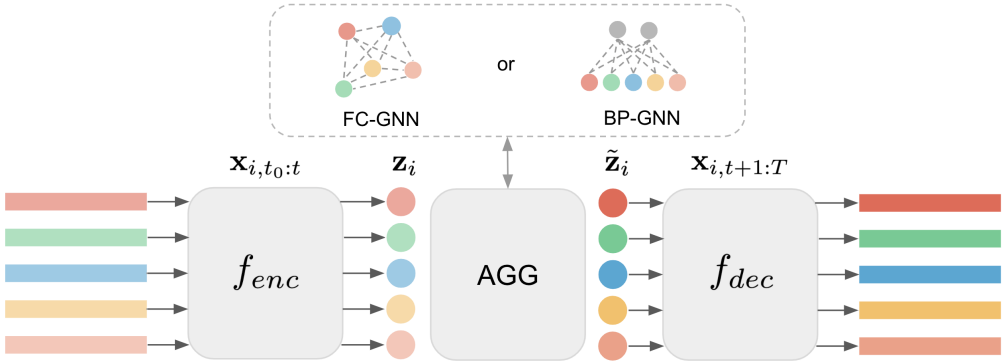


Figure 1: Illustration of the presented method under fully connected and bipartite graph assumptions.

4 METHOD

In Section 2.1 we discussed two different sorts of forecasting methods: Global Univariate Models and Multivariate Models. We cast our multivariate algorithm as a modular extension of the univariate case from Equation 1. We can break down the Univariate Model in two main steps $f_u = f_{enc} \circ f_{dec}$ such that $\mathbf{x}_{i,t_0:t} \xrightarrow{f_{\text{enc}}} \mathbf{z}_i \xrightarrow{f_{\text{dec}}} \hat{\mathbf{x}}_{i,t+1:T}$, where f_{enc} encodes the input signal into an embedding $\mathbf{z}_i$ and f_{dec} estimates the future signal from this embedding. In our method, we include a multivariate aggregation module AGG in between f_{enc} and f_{dec} that propagates information among nodes in the latent space $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$. This aggregation module takes as input the embedding $\mathbf{z}_i = f_{enc}(\mathbf{x}_{i,t_0:t})$, and outputs a modified embedding $\tilde{\mathbf{z}} = \text{AGG}(\mathbf{z})$ where information has been propagated among nodes. These new embeddings are then passed as input to the decoder f_{dec}. The resulting algorithm is:

$$\text{Univariate Encoder} \quad \mathbf{z}_i = f_{enc}(\mathbf{x}_{i,t_0:t}) \tag{4}$$

$$\text{Multivariate extension} \quad \tilde{\mathbf{z}} = \text{AGG}(\mathbf{z}) \tag{5}$$

$$\text{Univariate Decoder} \quad \hat{\mathbf{x}}_{i,t+1:T} = f_{dec}(\tilde{\mathbf{z}}_i) \tag{6}$$

Notice the overall model is multivariate but f_{enc} and f_{dec} remain univariate, allowing a modular extension from current univariate methods to multivariate. Additionally, in contrast to the most recent methods (Wu et al., 2020; Shang et al., 2021), our model does not propagate information among nodes at every time step $[t_0, \ldots, t]$ but only in the AGG module. This makes the algorithm potentially cheaper since the message propagation step is usually an expensive operation. Additionally, we experimented concatenating a unique identifier to each $\mathbf{z}_i$ at the input of the aggregation module AGG which resulted in a significant improvement in accuracy as we will show in the experiments section. In the following subsections we propose two different Graph Neural Network solutions for the AGG module, each has its benefits and weaknesses regarding performance and scalability.

Fully connected graph assumption | FC-GNN

From the two assumptions, this is the most straightforward to implement given a standard GNN. It results in good performance but its complexity scales $O(N^2)$ with respect to the number of nodes N. Here, we directly use the Graph Neural Network defined in Section 2.2 as the aggregation module AGG. We assume a fully connected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where each time series embedding $\mathbf{z}_i$ is associated to a node of the graph $i \in \mathcal{V}$, specifically, the embedding $\mathbf{z}_i$ is provided as the input $\mathbf{h}_i^0$ to the GNN from Equation 3, then the GNN runs for L layers and the output node embedding $\mathbf{h}_i^L$ is provided as the input $\tilde{\mathbf{z}}_i$ of the decoder. Moreover, despite the fully connected assumption, the GNN infers attention weights $\alpha_{ij} \in (0, 1)$ (eq. 3) for each edge and input sample. This is equivalent to dynamically inferring a latent graph where edges are associated to soft values in the range $(0, 1)$. In the experiments section, we name this model a Fully Connected Graph Neural Network (FC-GNN).

Bipartite graph assumption | BP-GNN

In the previous section we presented a GNN method that exchanges information among time series under a fully connected graph assumption which computationally scales $O(N^2)$. In this section we introduce a bipartite graph assumption which reduces the complexity to $O(NK)$, K being a

parameter of choice $K \ll N$. To accomplish this, we define a bipartite graph $\mathcal{G} = (\mathcal{Y}, \mathcal{U}, \mathcal{E})$ with two sets of nodes $\mathcal{Y}$ and $\mathcal{U}$. Here $\mathcal{Y}$ is a set of N nodes corresponding to the N dimensions of the multivariate time series and $\mathcal{U}$ is a set of K auxiliary nodes. Each time series embedding $\mathbf{z}_i$ is associated to one of the nodes $i \in \mathcal{Y}$. Additionally, we define embeddings $\mathbf{u} = \{\mathbf{u}_1, \dots \mathbf{u}_K\}$ associated to the auxiliary set $\mathcal{U}$. Edges $\mathcal{E}$ interconnect all nodes between the two subsets $\{\mathcal{Y}, \mathcal{U}\}$, but there are no connections among nodes of the same subset. This results in $2NK$ edges, with a computation complexity of $O(NK)$.

The algorithm works in the following way. We input into the GNN the union of the two node subsets $\mathcal{V} = \mathcal{Y} \cup \mathcal{U}$. Specifically, the input embedding $\mathbf{h}^0$ defined in Equation 3 is the concatenation of the time series embeddings $\mathbf{z}$ with the auxiliary node embeddings $\mathbf{u}$ (i.e. $\mathbf{h}^0 = \mathbf{z}||\mathbf{u}$) where $\mathbf{u}$ is initialized as Gaussian noise. Then, messages follow an asynchronous schedule, first they are sent from the time series nodes to the auxiliary nodes $\mathcal{Y} \rightarrow \mathcal{U}$, next the other way around $\mathcal{U} \rightarrow \mathcal{Y}$. Notice the auxiliary nodes $\mathcal{U}$ do not contain any information about the data until they have received messages from $\mathcal{Y}$, this is why we chose this schedule. A visual illustration of the whole algorithm is presented in Figure 1, where the bipartite graph is depicted at the top right of the image (BP-GNN).

We have conceptually defined the Bipartite Graph Neural Network. In Table 1 we introduce the equations that formally define it as an extension of the standard GNN Equation 3. Notice that it can be simply formulated as a two steps process where the indexes i, j belong to each one of the subsets $\mathcal{U}$ or $\mathcal{V}$ depending on the direction of the messages ($\mathcal{V} \rightarrow \mathcal{U}$ or $\mathcal{U} \rightarrow \mathcal{V}$). Additionally, we used different learnable parameters between Step 1 and Step 2 in the modules ϕ_e, ϕ_α and ϕ_h since it resulted in better accuracy than sharing parameters. Following, we define the adjacency matrices corresponding to the two message passing steps (assuming all inference parameters $\alpha_{ij} = 1$):

Step 1 $\mathcal{Y} \rightarrow \mathcal{U}$	Step 2 $\mathcal{U} \rightarrow \mathcal{Y}$
Equation 3 where $i \in \mathcal{U} \quad \& \quad j \in \mathcal{Y}$	Equation 3 where $i \in \mathcal{Y} \quad \& \quad j \in \mathcal{U}$

Table 1: BP-GNN formulation.

$$A_1 = \begin{vmatrix} 0_{N\times N} & 0_{N\times K} \\ 1_{K\times N} & 0_{K\times K} \end{vmatrix}, \qquad A_2 = \begin{vmatrix} 0_{N\times N} & 1_{N\times K} \\ 0_{K\times N} & 0_{K\times K} \end{vmatrix}, \qquad \tilde{A} = A_2 A_1 = \begin{vmatrix} K_{N\times N} & 0_{N\times K} \\ 0_{K\times N} & 0_{K\times K} \end{vmatrix} \quad (7)$$

The two adjacency matrices A_1 and A_2 are non-symmetric and they define a directed message passing scheme. A_1 refers to $\mathcal{Y} \rightarrow \mathcal{U}$ and A_2 refers to $\mathcal{U} \rightarrow \mathcal{Y}$. After running these two propagation steps, all time series nodes $\mathcal{Y}$ will have received messages from all other time series nodes $\mathcal{Y}$ as in the fully connected graph case, but the number of messages have been reduced (iff $0 \leq K < N/2$) due to the factorization of the adjacency matrix. The first N rows and columns in matrix $\tilde{A}$ from Equation 7 represent the sum of all paths that communicate the time series nodes $\mathcal{Y}$ among them after the two updates $A_2 A_1$.

Architecture details

As explained in this Section 4, our method is composed of three main modules, the encoder f_{enc}, the decoder f_{dec} and the aggregation function AGG. We choose to use relatively simple networks as encoder and decoder. The decoder f_{dec} is defined as a Multi Layer Perceptron (MLP) with a single hidden layer and a residual connection in all experiments. The encoder f_{enc} is also defined as an MLP for METR-LA, PEMS-BAY and our synthetic datasets and as a Convolutional Neural Network (CNN) for the other datasets since these require a larger encoding length and the translation equivariance of CNNs showed to be more beneficial. The combination of these simple networks with our proposed aggregation module fully defines our model. Further details are in Appendix A.1.

The a aggregation module AGG was defined under two different assumptions in, the Fully Connected Graph assumption (FC-GNN) and the Bipartite Graph assumption (BG-GNN). In both cases the GNN is fully parametrized by the networks ϕ_e, ϕ_h and ϕ_α. ϕ_e consists of a two layers MLP, ϕ_h is a one layer MLP with a skip connection from he input to the output and ϕ_α is just a linear layer followed by a Sigmoid activation function. All these architecture choices are explained in more detail in Appendix A.1. The loss is computed from the estimated values $\hat{\mathbf{x}}_{t+1:T}$ from Equation 6 and the ground truth as $\mathcal{L} = l(\hat{\mathbf{x}}_{t+1:T}, \mathbf{x}_{t+1:T})$. We used the Mean Absolute Error as the loss function $\mathcal{L}$ in all our experiments.

5 EXPERIMENTS

5.1 DATASETS AND BASELINES

We first evaluate our method in METR-LA and PEMS-BAY datasets from (Li et al., 2017) which record traffic speed statistics on the highways of Los Angeles county and the Bay Area respectively. We also evaluate in the publicly available Solar-Energy, Traffic, Electricity and Exchange-Rate. The specifications for each dataset are presented in Table 2.

	#Nodes	# Samples	Enc. length	Dec. length
METR-LA	207	34,272	12	12
PEMS-BAY	325	52,116	12	12
Solar-Energy	137	52,560	168	1
Traffic	862	17,544	168	1
Electricity	321	26,304	168	1
Exchange-Rate	8	7,588	168	1

Table 2: Dataset specifications.

We compare to a variety of baselines including previous works and variations of our proposed method. We distinguish the following three main types of baselines (Univariate / Multivariate with known graph / Multivariate with graph inference):

- Univariate: In this case a single model is trained for all time series but they are treated independently without message exchange among them. These baselines include a simple linear Auto Regressive model (AR) and a variation of our FC-GNN method that we called NE-GNN where all edges have been removed such that the aggregation function AGG becomes equivalent to a multilayer perceptron defined by ϕ_h in Equation 3.
- Multivariate with a known graph: These methods require a previously defined graph, therefore they are restricted to those datasets where an adjacency matrix can be pre-defined (e.g. METR-LA, PEMS-BAY). From this group we compare to DCRNN (Li et al., 2017), STGCN (Yu et al., 2017) and MRA-BGCN (Chen et al., 2020a).
- Multivariate with graph inference or attention: These methods exchange information among different time series by simultaneously inferring relations among them or by attention mechanisms. From this group we compare to LDS (Franceschi et al., 2019), LST-Skip (Lai et al., 2018), TPA-LSTM (Shih et al., 2019), MTGNN (Wu et al., 2020) and GTS (Shang et al., 2021). Graph WaveNet (Wu et al., 2019) also belongs to this group but unlike the others it jointly uses a pre-defined adjacency matrix. We additionally include a variation of our FC-GNN without unique node ids, we denote it by *(w/o id)* beside the model name. GTS numbers have been obtained by averaging over 3 runs its official implementation. https://github.com/chaoshangcs/GTS.

5.2 MAIN RESULTS

In this section we evaluate our method in METR-LA and PEMS-BAY datasets. For this experiment we used the training setup from GTS (Shang et al., 2021) which uses the dataset partitions and evaluation metrics originally proposed in (Li et al., 2017). The model has been trained by minimizing the Mean Absolute Error (MAE) between the predicted and the ground truth samples. The reported metrics are MAE, Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) from (Li et al., 2017). All metrics have been averaged over 5 runs. All our models (FC-GNN, BP-GNN and NE-GNN) contain 2 graph convolutional layers, 64 features in the hidden layers, Swish activation functions (Ramachandran et al., 2017) and have been trained with batch size 16. The number of auxiliary nodes for BP-GNN was set to $K = 4$. Time experiments report the average forward pass in seconds for a batch size 16 in a Tesla V100-SXM GPU. Further implementation details are provided in Appendix B.

Results are reported in tables 3 and 4. FC-GNN outperforms other methods in most metrics while being computationally cheaper than previous works. On the other hand, BP-GNN performs very competitively w.r.t. previous works (even outperforming all previous methods in some metrics) but with a vast decrease in computation. Furthermore, these performances are achieved without providing the structure information of the city (i.e. pre-defined adjacency) unlike in those methods that require it or in Graph Wavenet that optionally uses it. Additionally, notice the performance gap between FC-GNN and NE-GNN is larger when including a unique identifier of the nodes. This means the network can better leverage the information exchange among nodes when they are uniquely identified,

	METR-LA									PEMS-BAY								
	15 min			30 min			60 min			15 min			30 min			60 min		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
Linear / AR	3.81	8.80	9.13%	4.94	11.14	12.17%	6.30	12.91	16.72%	1.59	3.41	3.27%	2.15	4.87	4.77 %	2.97	6.65	7.03%
DCRNN	2.77	5.38	7.30%	3.15	6.45	8.80%	3.60	7.60	10.50%	1.38	2.95	2.90%	1.74	3.97	3.90%	2.07	4.74	4.90%
STGCN	2.88	5.74	7.6%	3.47	7.24	9.6%	4.59	9.40	12.7%	1.36	2.96	2.9%	1.81	4.27	4.2%	2.49	5.69	5.8%
MRA-BGCN	2.67	**5.12**	**6.8%**	3.06	6.17	8.3%	3.49	7.30	10.0%	**1.29**	**2.72**	2.9%	**1.61**	**3.67**	3.8%	**1.91**	**4.46**	4.6%
Graph WaveNet*	2.69	5.15	6.90%	3.07	6.22	8.37%	3.53	7.37	10.01%	1.30	2.74	**2.73%**	1.63	3.70	**3.67%**	1.95	4.52	4.63%
LDS	2.75	5.35	7.1%	3.14	6.45	8.6%	3.63	7.67	10.34%	1.33	2.81	2.8%	1.67	3.80	3.8%	1.99	4.59	4.8 %
MTGNN	2.69	5.18	6.86%	3.05	6.17	8.19%	3.49	7.23	9.87%	1.32	2.79	2.77%	1.65	3.74	3.69%	1.94	4.49	**4.53%**
GTS	2.64	5.19	6.79%	3.06	6.30	8.24%	3.56	7.55	9.95%	1.35	2.84	2.85%	1.67	3.82	3.80%	1.96	4.53	4.62%
Ablation study																		
NE-GNN *(w/o id)*	2.80	5.73	7.44%	3.40	7.15	9.69%	4.22	8.79	12.90%	1.40	3.02	2.92 %	1.85	4.24	4.18 %	2.38	5.47	5.91%
FC-GNN *(w/o id)*	2.77	5.66	7.34%	3.36	7.05	9.54%	4.14	8.62	12.63%	1.39	3.01	2.87%	1.82	4.19	4.09%	2.32	5.35	5.72%
NE-GNN	2.69	5.53	7.25%	3.13	6.71	9.01%	3.61	7.85	11.01%	1.35	2.88	2.82%	1.71	3.88	3.84 %	2.06	4.74	4.92%
Our models																		
FC-GNN	**2.60**	5.19	**6.78%**	**2.95**	**6.15**	**8.14%**	**3.35**	**7.14**	**9.73%**	1.33	2.82	2.79%	1.65	3.75	**3.72%**	1.93	**4.46**	**4.53%**
BP-GNN (K=4)	2.64	5.38	7.00%	3.02	6.42	8.47%	3.41	7.33	9.88%	1.33	2.82	2.80%	1.66	3.78	3.75%	1.94	**4.46**	4.57%

Table 3: Benchmark on METR-LA and PEMS-BAY datasets. Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) are reported for different time horizons {15, 30, 60} minutes. Results have been averaged over 5 runs.

but it still benefits from the message passing scheme when they are not uniquely identified (w/o id) thanks to the dynamical inference. Time results are presented in Table 4. BP-GNN is the most efficient algorithm in both METR-LA and PEMS-BAY by a large margin. FC-GNN is also more efficient than previous methods in both datasets but it is still limited by the $O(N^2)$ scalability. In Section 5.3 we will evaluate in a larger dataset showing that BP-GNN is even more efficient w.r.t. to other methods as the graph becomes larger thanks to its linear scalability $O(N)$. In this timing benchmark (Table 4) we included all methods from the previous Table 3 that have a publicly available implementation in METR-LA and PEMS-BAY datasets.

	Forward Time (s)	
	METR-LA	PEMS-BAY
Linear	.0002	.0002
DCRNN	.2559	.2754
Graph WaveNet	.0500	.0673
MTGNN	.0160	.0371
GTS	.0869	.1087
NE-GNN	.0033	.0047
FC-GNN	.0108	.0253
BP-GNN (K=4)	.0044	.0046

Table 4: Forward time in seconds for different methods.

5.3 SINGLE STEP FORECASTING

In this section, we evaluate our method in the publicly avilable Solar-Energy, Traffic, Electricity and Exchange-Rate datasets. In contrast to METR-LA and PEMS-BAY, these datasets do not contain spatial information from which a graph can be pre-defined, therefore, methods that rely on a known graph are not directly applicable. We use the same training settings as (Lai et al., 2018; Shih et al., 2019; Wu et al., 2020) firstly proposed by TLST-skip (Lai et al., 2018). In this experiment the network is trained to predict only one time step into the future (single step forecasting) with a given horizon (3, 6, 12, or 24) by minimizing the Mean Absolute Error (MAE). All datasets have been split in 60%/20%/20% for training/val/test respectively. As proposed in (Lai et al., 2018) we use the Root Relative Squared Error (RSE) and Empirical Correlation Coefficient (CORR) as evaluation metrics (both defined in Appendix C.2). All our models (FC-GNN, BP-GNN and NE-GNN) contain 2 graph convolutional layers and 128 features in the hidden layers. The timing results have been run with batch size 16. Further implementation details are provided in Appendix C.1.

Results in Table 5 are consistent with the previous experiment. FC-GNN outperforms all previous works in most metrics. There is a significant improvement w.r.t. its cousin baseline NE-GNN, this demonstrates that sharing information among nodes is beneficial (except in Exchange-Rate dataset which only contains 8 nodes). BP-GNN performs better than previous methods in most metrics and very close to FC-GNN. Regarding the timing results (Table 6), BP-GNN is the most efficient graph inference method by a large margin in most datasets. The larger the number of nodes in the dataset, the larger the computational improvement of BP-GNN w.r.t. other methods due to its linear scalability.

Dataset		Solar-Energy				Traffic				Electricity				Exchange-Rate			
		Horizon				Horizon				Horizon				Horizon			
Methods	Metrics	3	6	12	24	3	6	12	24	3	6	12	24	3	6	12	24
AR	RSE	.2435	.3790	.5911	.8699	.5991	.6218	.6252	.6293	.0995	.1035	.1050	.1054	.0228	.0279	.0353	.0445
	CORR	.9710	.9263	.8107	.5314	.7752	.7568	.7544	.7519	.8845	.8632	.8591	.8595	.9734	.9656	.9526	.9357
RNN-GRU	RSE	.1843	.2559	.3254	.4643	.4777	.4893	.4950	.4973	.0864	.0931	.1007	.1007	.0226	.0280	.0356	.0449
	CORR	.9843	.9690	.9467	.8870	.8721	.8690	.8614	.8588	.9283	.9135	.9077	.9119	.9735	.9658	.9511	.9354
LST-skip	RSE	.1843	.2559	.3254	.4643	.4777	.4893	.4950	.4973	.0864	.0931	.1007	.1007	.0226	.0280	.0356	.0449
	CORR	.9843	.9690	.9467	.8870	.8721	.8690	.8614	.8588	.9283	.9135	.9077	.9119	.9735	.9658	.9511	.9354
TPA-LSTM	RSE	.1803	.2347	.3234	.4389	.4487	.4658	.4641	.4765	.0823	.0916	.0964	.1006	**.0174**	**.0241**	.0341	**.0444**
	CORR	.9850	.9742	.9487	.9081	.8812	.8717	.8717	.8629	.9439	.9337	.9250	.9133	**.9790**	**.9709**	**.9564**	**.9381**
MTGNN	RSE	.1778	.2348	.3109	.4270	.4162	.4754	**.4461**	**.4535**	.0745	**.0878**	**.0916**	**.0953**	.0194	.0259	.0349	.0456
	CORR	.9852	.9726	.9509	.9031	.8963	.8667	**.8794**	**.8810**	.9474	.9316	.9278	.9234	**.9786**	**.9708**	**.9551**	**.9372**
NE-GNN	RSE	.1898	.2580	.3472	.4441	.4212	.4586	.4679	.4743	.0762	.0917	.0966	.0994	.0175	**.0244**	**.0338**	.0447
	CORR	.9829	.9663	.9367	.8905	.8951	.8748	.8700	.8670	.9494	.9362	.9308	.9262	.9769	.9686	.9535	.9352
FC-GNN	RSE	**.1651**	**.2202**	**.2981**	**.3997**	**.4057**	**.4395**	**.4624**	**.4620**	**.0732**	.0907	**.0915**	**.0979**	**.0174**	.0245	.0344	.0450
	CORR	**.9876**	**.9765**	**.9551**	**.9148**	**.9024**	**.8850**	**.8764**	**.8751**	**.9521**	**.9404**	**.9351**	**.9294**	.9772	.9685	.9538	.9349
BP-GNN (K=4)	RSE	**.1704**	**.2257**	**.3072**	**.4050**	**.4095**	**.4470**	.4640	.4641	**.0740**	**.0898**	.0940	.0980	.0175	**.0244**	**.0339**	**.0442**
	CORR	**.9865**	**.9751**	**.9522**	**.9138**	**.8999**	**.8820**	.8744	.8723	**.9519**	**.9396**	**.9345**	**.9288**	.9769	.9684	.9530	.9360

Table 5: Benchmark on Solar-Energy, Traffic, Electricity and Exchange-Rate. Root Relative Squared Error (RSE) and Empirical Correlation Coefficient (CORR) are reported for different horizons {3, 6, 12, 24}. All results have been averaged over 5 runs.

	Exchange	Solar	Electricity	Traffic
# Nodes	8	137	321	862
MTGNN	.0062	.0146	.0771	1.1808
NE-GNN	.0034	.0059	.0067	.0117
FC-GNN	.0053	.0109	.0536	.4184
BP-GNN	.0076	.0076	.0084	.0121

Table 6: Forward time in seconds for different methods in Solar-Energy, Traffic, Electricity and Exchange-Rate datasets.

For example, in Solar-Energy (137 nodes), BP-GNN is 1.43 times faster than FC-GNN and 1.92 times faster than MTGNN. On the other hand, in a larger dataset as Electricity (321 nodes) BP-GNN is 6.38 times faster than FC-GNN and 9.18 times faster than MTGNN. In the largest dataset, Traffic (862 nodes), BP-GNN becomes 34.58 times faster than FC-GNN and 97.59 times faster than MTGNN, although in traffic, MTGNN and FC-GNN did not fit in the GPU for a batch of 16 (also because to the $O(N^2)$ complexity), and we had to pass the samples in batches of 2 which eliminates part of the GPU parallelization. Running BP-GNN in batches of 2, would result in it being 7.71 and 18.92 times faster than FC-GNN and MTGNN respectively. Finally, in such small graphs as Exchange Rate, there is no computational benefit in using the bipartite assumption since the number of edges for both the bipartite and the fully connected graphs becomes the same (for K=4 and N=8).

5.4 Inferred graph analysis

In this section we study the adjacency matrices inferred by our FC-GNN and BP-GNN methods. For this purpose, we created two synthetic datasets of $N = 10$ time series and $T = 10.000$ timesteps each. (a) "Cycle Graph Gaussians" samples each series value $\mathbf{x}_{i,t}$ from the past $(t-5)$ of another series $(i-1 \mod N)$ from the panel. The resulting adjacency matrix is a directed cycle graph. More formally, the dataset is generated from the following gaussian distribution $\mathbf{x}_{i,t} \sim \mathcal{N}(\beta \mathbf{x}_{i-1 \mod N, t-5}; \sigma^2)$, where $\beta = 0.9$ and $\sigma = 0.5^2$. (b) We call the second dataset "Noisy Sinusoids", inspired by the Discrete Sine Transformation we generate arbitrary signals as the sum of different sinusoids plus gaussian noise. We share the same sinusoids among the first half $1 \leq i \leq 5$ of the panel and among the second half $6 \leq i \leq 10$ while sampling different gaussian noise for each signal i. This creates strong dependencies among the signals within each half. The expected adjacencies are plotted in Figure 2. Further details and visualizations of these two datasets are provided in Appendix D.1. Both BP-GNN and FC-GNN consist of a single graph convolution from which adjacencies are obtained. We predict the next time step into the future and optimize the MAE during training. Since our graph inference mechanism is dynamic we average them over 10 timesteps t. For BP-GNN we construct the adjacencies as the first N rows and columns of $\tilde{A} = A_2 A_1$ (Eq. 7, Sec. 4). Notice that small K values in BP-GNN may enforce sharing auxiliary nodes and in consequence a more dense adjacency $\tilde{A}$. Further implementation details are described in Appendix D.2.

Results are reported in Figure 2. FC-GNN perfectly matches the ground truth adjacencies and it has the lowest MAE test loss. In the "Cycle Graph Gaussians" dataset, BP-GNN infers a similar matrix to

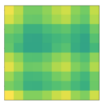
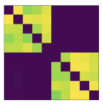
		Expected Adjacency	NE-GNN	FC-GNN	BP-GNN K=10	BP-GNN K=5	BP-GNN K=2	BP-GNN K=1
Cycle Graph Gaussian	Adjacency							
	MAE	-	0.9291	0.4023	0.4098	0.4149	0.4295	0.4955
Noisy Sinusoids	Adjacency							
	MAE	-	0.2149	0.1868	0.1873	0.1878	0.1881	1.892

Figure 2: Inferred adjacency matrices and MAE losses in the proposed synthetic datasets.

the ground truth when provided with enough auxiliary nodes $K = 10$, but the matrix becomes denser when reducing K, which is expected since different messages are forced to share the same auxiliary nodes. Despite that, in both datasets, BP-GNN manages to obtain very competitive performance (even for small K) by inferring suboptimal matrices with denser connections. In the "Noisy Sinusoids" dataset, the benefit of increasing K is smaller, this is coherent with the fact the ground truth matrix is denser. In all cases the proposed models significantly outperform their non-edge variant NE-GNN. Further visualizations for different random seeds are provided in Appendix D.3.

5.5 Choosing the number of auxiliary nodes K

In the previous section we evaluated BP-GNN for different K values, and we found small values were enough to achieve good performance in the denser adjacency dataset (Noisy Sinusoids). In real world scenarios, we found a similar behavior, where results are competitive with small K values. Therefore, we chose a relatively small number of auxiliary nodes $K = 4$ that resulted in a good trade-off accuracy vs computation. In Figure 3, we plot the MAE for different K values in METR-LA for a 15 min time horizon, and we see the best performances are already achieved at a small K values. In the middle plot of Figure 3, we report the forward running time in (s) for a batch size 16 for METR-LA as we sweep different K values. Finally, the right plot of Figure 3 shows the scalability of the number of edges w.r.t. to the number of nodes for $O(N^2)$ (e.g. FC-GNN) and $O(2N4)$ (i.e. BP-GNN (K=4). This illustrates the vast difference in scalability as the number of nodes increases.

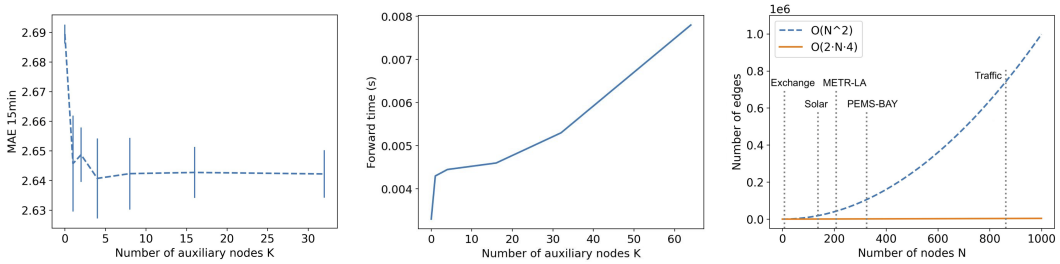


Figure 3: (Left) Mean Absolute Error in METR-LA for different K values (10 runs average). (Middle) Running times in METR-LA. (Right) Number of edges w.r.t to the number of nodes.

6 Conclusions

We presented a novel approach for multi-variate time series forecasting which lends itself to easy integration into existing univariate approaches by ways of a graph neural network (GNN) component. This GNN infers a latent graph in the space of the embeddings of the univariate time series and can be inserted in many neural-network based forecasting models. We show that this additional graph/dependency structure improves forecasting accuracy both in general and in particular when compared to the state of the art. We alleviate typical scalability concerns for GNNs via allowing the introduction of auxiliary nodes in the latent graph which we construct as a bipartite graph that bounds the computational complexity. We show that this leads to computationally favorable properties as expected while the trade-off with forecasting accuracy is manageable.

REFERENCES

Andersen, T. G., Bollerslev, T., Christoffersen, P., and Diebold, F. X. (2005). Volatility forecasting.

Bandara, K., Shi, P., Bergmeir, C., Hewamalage, H., Tran, Q., and Seaman, B. (2019). Sales demand forecast in e-commerce using a long short-term memory neural network methodology. In *International Conference on Neural Information Processing*, pages 462–474. Springer.

Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, B., Maddix, D. C., Türkmen, A. C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Callot, L., and Januschowski, T. (2020). Neural forecasting: Introduction and literature overview. *CoRR*, abs/2004.10240.

Böse, J.-H., Flunkert, V., Gasthaus, J., Januschowski, T., Lange, D., Salinas, D., Schelter, S., Seeger, M., and Wang, Y. (2017). Probabilistic demand forecasting at scale. *Proceedings of the VLDB Endowment*, 10(12):1694–1705.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Chen, W., Chen, L., Xie, Y., Cao, W., Gao, Y., and Feng, X. (2020a). Multi-range attentive bicomponent graph convolutional network for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3529–3536.

Chen, Y., Kang, Y., Chen, Y., and Wang, Z. (2020b). Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing*, 399:491–501.

Cranmer, M., Sanchez-Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., and Ho, S. (2020). Discovering symbolic models from deep learning with inductive biases. *arXiv preprint arXiv:2006.11287*.

de Bézenac, E., Rangapuram, S. S., Benidis, K., Bohlke-Schneider, M., Kurle, R., Stella, L., Hasson, H., Gallinari, P., and Januschowski, T. (2020). Normalizing kalman filters for multivariate time series analysis. *Advances in Neural Information Processing Systems*, 33.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852.

Deng, A. and Hooi, B. (2021). Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence, Vancouver, BC, Canada*, pages 2–9.

Eisenach, C., Patel, Y., and Madeka, D. (2020). Mqtransformer: Multi-horizon forecasts with context dependent and feedback-aware attention. *arXiv preprint arXiv:2009.14799*.

et al., F. P. (2020). Forecasting: theory and practice. *arXiv preprint arXiv:2012.03854*.

Franceschi, L., Niepert, M., Pontil, M., and He, X. (2019). Learning discrete structures for graph neural networks. In *International conference on machine learning*, pages 1972–1982. PMLR.

Fuchs, F. B., Worrall, D. E., Fischer, V., and Welling, M. (2020). Se (3)-transformers: 3d roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503*.

Garcia, V. and Bruna, J. (2017). Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*.

Geweke, J. (1977). The dynamic factor analysis of economic time series. *Latent variables in socio-economic models*.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR.

Hyndman, R. J. and Athanasopoulos, G. (2017). Forecasting: Principles and practice. *www. otexts. org/fpp.*, 987507109.

Januschowski, T., Gasthaus, J., Wang, Y., Salinas, D., Flunkert, V., Bohlke-Schneider, M., and Callot, L. (2019). Criteria for classifying forecasting methods. *International Journal of Forecasting*.

Kaushik, S., Choudhury, A., Sheron, P. K., Dasgupta, N., Natarajan, S., Pickett, L. A., and Dutt, V. (2020). Ai in healthcare: time-series forecasting using statistical, neural, and ensemble architectures. *Frontiers in big data*, 3:4.

Kazi, A., Cosmo, L., Navab, N., and Bronstein, M. (2020). Differentiable graph module (dgm) for graph convolutional networks. *arXiv preprint arXiv:2002.04999*.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. (2018). Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697. PMLR.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kossen, J., Band, N., Lyle, C., Gomez, A. N., Rainforth, T., and Gal, Y. (2021). Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. *arXiv preprint arXiv:2106.02584*.

Kurle, R., Rangapuram, S. S., de Bézenac, E., Günnemann, S., and Gasthaus, J. (2020). Deep rao-blackwellised particle filters for time series forecasting. *Advances in Neural Information Processing Systems*, 33.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. (2018). Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104.

Larson, P. D. (2001). Designing and managing the supply chain: concepts, strategies, and case studies. *Journal of Business Logistics*, 22(1):259.

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*, volume 32. Curran Associates, Inc.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. (2017). Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

Liberty, E., Karnin, Z., Xiang, B., Rouesnel, L., Coskun, B., Nallapati, R., Delgado, J., Sadoughi, A., Astashonok, A., Das, P., Balioglu, C., Charkravarty, S., Jha, M., Gaultier, P., Januschowski, T., Flunkert, V., Wang, B., Gasthaus, J., Rangapuram, S., Salinas, D., Schelter, S., Arpin, D., and Smola, A. (2020). Elastic machine learning algorithms in Amazon SageMaker. In *Proceedings of the 2020 International Conference on Management of Data*, SIGMOD '20, New York, NY, USA. ACM.

Lim, B., Arık, S. Ö., Loeff, N., and Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*.

Lim, B. and Zohren, S. (2021). Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200209.

Lütkepohl, H. (2005). *New introduction to multiple time series analysis*. Springer Science & Business Media.

Montero-Manso, P. and Hyndman, R. J. (2022). Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*.

Mudelsee, M. (2019). Trend analysis of climate time series: A review of methods. *Earth-science reviews*, 190:310–322.

Petropoulos, F., Apiletti, D., Assimakopoulos, V., Babai, M. Z., Barrow, D. K., Bergmeir, C., Bessa, R. J., Boylan, J. E., Browell, J., Carnevale, C., Castle, J. L., Cirillo, P., Clements, M. P., Cordeiro, C., Oliveira, F. L. C., Baets, S. D., Dokumentov, A., Fiszeder, P., Franses, P. H., Gilliland, M., Gönül, M. S., Goodwin, P., Grossi, L., Grushka-Cockayne, Y., Guidolin, M., Guidolin, M., Gunter, U., Guo, X., Guseo, R., Harvey, N., Hendry, D. F., Hollyman, R., Januschowski, T., Jeon, J., Jose, V. R. R., Kang, Y., Koehler, A. B., Kolassa, S., Kourentzes, N., Leva, S., Li, F., Litsiou, K., Makridakis, S., Martinez, A. B., Meeran, S., Modis, T., Nikolopoulos, K., Önkal, D., Paccagnini, A., Panapakidis, I., Pavía, J. M., Pedio, M., Pedregal, D. J., Pinson, P., Ramos, P., Rapach, D. E., Reade, J. J., Rostami-Tabar, B., Rubaszek, M., Sermpinis, G., Shang, H. L., Spiliotis, E., Syntetos, A. A., Talagala, P. D., Talagala, T. S., Tashman, L., Thomakos, D., Thorarinsdottir, T., Todini, E., Arenas, J. R. T., Wang, X., Winkler, R. L., Yusupova, A., and Ziel, F. (2020). Forecasting: theory and practice. *arXiv preprint arXiv:2012.03854*.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. (2018). Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, pages 7785–7794.

Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. (2021). Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. *Proceedings of ICLR 2021*.

Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U., and Vollgraf, R. (2020). Multi-variate probabilistic time series forecasting via conditioned normalizing flows. *arXiv preprint arXiv:2002.06103*.

Roberts, S., Osborne, M., Ebden, M., Reece, S., Gibson, N., and Aigrain, S. (2013). Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110550.

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., and Gasthaus, J. (2019). High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *Advances in Neural Information Processing Systems 32*.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191.

Satorras, V. G., Hoogeboom, E., and Welling, M. (2021). E (n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*.

Seo, Y., Defferrard, M., Vandergheynst, P., and Bresson, X. (2018). Structured sequence modeling with graph convolutional recurrent networks. In *International Conference on Neural Information Processing*, pages 362–373. Springer.

Shang, C., Chen, J., and Bi, J. (2021). Discrete graph structure learning for forecasting multiple time series. *arXiv preprint arXiv:2101.06861*.

Shih, S.-Y., Sun, F.-K., and Lee, H.-y. (2019). Temporal pattern attention for multivariate time series forecasting. *Machine Learning*, 108(8):1421–1441.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, Y., Smola, A., Maddix, D., Gasthaus, J., Foster, D., and Januschowski, T. (2019a). Deep factors for forecasting. In *International Conference on Machine Learning*, pages 6607–6617.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019b). Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12.

Wen, R., Torkkola, K., Narayanaswamy, B., and Madeka, D. (2017). A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*.

Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., and Zhang, C. (2020). Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 753–763.

Wu, Z., Pan, S., Long, G., Jiang, J., and Zhang, C. (2019). Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121*.

Yu, B., Yin, H., and Zhu, Z. (2017). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.

Zhang, R., Hao, Y., Yu, D., Chang, W.-C., Lai, G., and Yang, Y. (2020). Correlation-aware change-point detection via graph neural networks. In *International Conference on Neural Information Processing*, pages 555–567. Springer.

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 Transactions on Intelligent Transportation Systems*, 21(9):3848–3858.