Adaptive Graph Recurrent Network for Multivariate Time Series Imputation

Yakun Chen, Zihao Li, Chao Yang, Xianzhi Wang ✉,
Guodong Long, and Guandong Xu

University of Technology Sydney, Sydney NSW 2007, Australia
{yakun.chen,zihao.li,chao.yang}@student.uts.edu.au,
{xianzhi.wang,guodong.long,guandong.xu}@uts.edu.au

Abstract. Multivariate time series inherently involve missing values for various reasons, such as incomplete data entry, equipment malfunctions, and package loss in data transmission. Filling missing values is important for ensuring the performance of subsequent analysis tasks. Most existing methods for missing value imputation neglect inter-variable relations in time series. Although graph-based methods can capture such relations, the design of graph structures commonly requires domain knowledge. In this paper, we propose an adaptive graph recurrent network (AGRN) that combines graph and recurrent neural networks for multivariate time series imputation. Our model can learn variable- and time-specific dependencies effectively without extra information such as domain knowledge. Our extensive experiments on real-world datasets demonstrate our model's superior performance to state-of-the-art methods.

Keywords: Graph neural network · Multivariate time series imputation · Spatio-temporal graph learning

1 Introduction

Multivariate time series data is ubiquitous and has many applications in different fields, such as financial market [14], traffic flow [11] and industrial systems [25]. Due to some inevitable reasons, missing values likely appear in time series datasets. Taking the industrial environment as an example, accidents such as connection loss and hardware damage make missing values commonly seen in the collected data [20]. A direct and well-known method is to delete observations with missing values and just analyze the remaining part of the data. However, in some scenarios, the proportion of missing observations exceeds 80% [9]. Simply dropping missing values could cause serious information loss, which will harm the downstream data analysis task, such as classification and forecasting [7]. Different from time series forecasting, which aims to predict the future time steps based on previously recorded data, the position of the missing values is unpredictable, requiring the imputation model to harness known time steps to fill missing values (illustrated in Figure 1).

Existing research has employed statistics [22], machine learning- [2], and deep learning-based [16] methods to solve the imputation problem. Yet, they

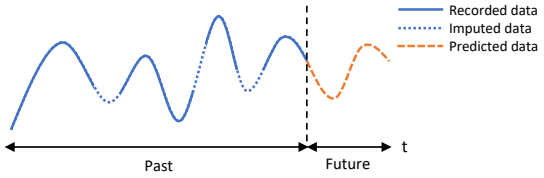

Fig. 1. An illustration of the difference between time series imputation and forecasting.

still face significant challenges in capturing dynamic spatio-temporal dependencies. Specifically, statistic and machine learning methods require time series data to be high-structured and follow their model assumptions. Deep learning-based methods [3, 4] simply apply Recurrent Neural Networks (RNNs) without considering variable dependencies for imputation tasks. Graph-based methods [6, 12] can capture spatial relations at the variable level, but they generally use pre-defined graph structures and thus cannot generalize well to more datasets.

In this paper, we propose an adaptive graph recurrent network (AGRN) for multivariate time series imputation. Instead of relying on pre-defined graphs [6, 12], our model can learn and refine variables relations only from data and use the learned graph to obtain variable- and time-specific dependencies, supporting filling missing values. Our contributions are summarized as follows: (1) We propose an adaptive graph recurrent network that combines graph convolution network and recurrent neural network for multivariate time series imputation; (2) Our graph learning module can automatically learn inter-variable relations without requiring domain knowledge. It improves the model's generality by dynamically adjusting graph edges during training; (3) Our extensive experiments on real-world datasets (air quality and traffic) show our model outperforms state-of-the-art models in multivariate time series imputation.

2 Related Work

Missing values have been a standing challenge in time series analysis, attracting lots of effort to solving this problem [7, 21]. Traditional approaches to time series imputation include statistical and machine learning-based methods. Autoregressive methods, such as Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA), can automatically fit their models to known data and generally obtain better results [22]. More advanced methods include Multivariate Imputation by Chained Equations (MICE) [1] and Variational Autoencoder (VAE)-based methods. The former uses chained equations to iteratively estimate each missing variable. GP-VAE [8], an example of the latter, conducts missing value imputation by mapping time series data to a latent space. Typical machine learning methods for time series imputation include k-nearest neighbors (kNN) [2], Expectation Maximization (EM) [19], and Matrix Factorization (MF) [5]. Such methods generally make strong assumptions (e.g.,

low-rankness and hypothetical distribution) about time series data, which limit their generalization ability.

Deep learning methods have been introduced to multivariate time series imputation, given their proven success in multiple applications, such as computer vision, speech processing, and natural language processing. Most existing methods are based on RNNs, Generative Adversarial Networks (GANs), and their variants. For example, GRU-D [4] applies a decay controller to the hidden states of Gated Recurrent Units (GRUs) for imputation. BRITS [3] employs a bidirectional RNN-based model to predict multiple correlated missing values in time series. In particular, adversarial network-based methods are generally good at reconstructing sequential data [15, 16, 18, 24]. SSGAN [18] uses a semi-supervised classifier and the temporal reminder matrix to learn data distribution to impute unlabeled time series data. While bearing their own advantages, those methods commonly lack the capability to take into account both spatio-temporal dependencies when filling missing values. Graph Neural Networks (GNNs) have recently been applied to multivariate time series imputation to overcome the above limitations [6, 12]. As an example, STGNN-DAE [12] leverages the power grid topology and time series data obtained from each meter in the grid to account for both spatial and temporal correlations. Another recent work is GRIN [6], which designs a spatial-temporal encoder to combine variable relations and time dependencies. Despite promising, all the above GNN-based methods require domain knowledge and explicit variable relations to generate the graph structure, thus introducing extra inductive bias and making their models less transferable. All the above-unresolved challenges motivate this paper.

3 Methodology

A multivariate time series imputation task takes as the input time series data $\mathbf{X} \in R^{N \times T}$, where N, T denote the number of variables and the number of time steps, respectively. A mask matrix $\mathbf{M} \in \{0, 1\}^{N \times T}$ indicates the locations of missing values in the time series, where $m_{n,t} = 0$ indicates $x_{n,t}$ is missing; otherwise, $m_{n,t} = 1$. The task's output $\widehat{\mathbf{Y}} \in R^{N \times T}$ bears the same dimensions as the input, with all the missing values filled up. As such, the task of multivariate time series imputation aims to determine the closest values to the underlying ground truth to fill the missing values in $\mathbf{X}$.

Our proposed framework (Figure 2) comprises four components: graph learning, graph convolution, spatio-temporal fusion, and prediction. It works as follows. First, the graph learning module uses the input signals to generate a graph representing variables relations. Then, the graph convolution module generates aggregated node representations with neighbor information by combining the raw input and the graph's adjacency matrix. Following that, the spatio-temporal fusion module employs Gated Recurrent Units (GRUs) for temporal information passing. Lastly, the prediction module fuses the outputs of the forward and backward branches to finally accomplish the missing value imputation.

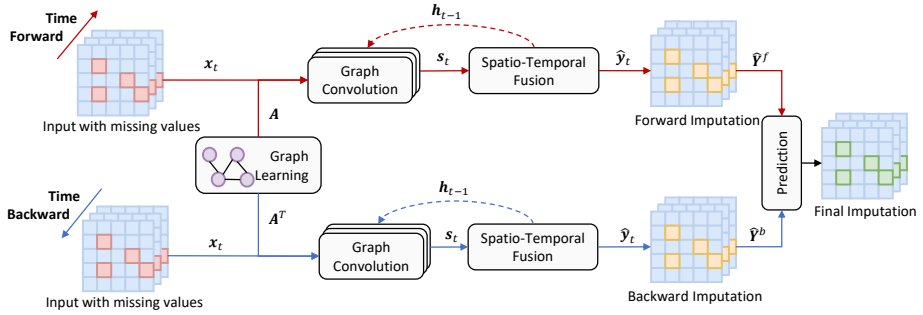

Fig. 2. The architecture of our proposed model.

3.1 Graph Learning Module

Graph Structure We denote the relations among all variables via a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the set of nodes and edges respectively. For an edge $e_{ij} \in \mathcal{E}$, it could be represented as an ordered tuple (v_i, v_j) which means the edge from node v_i to v_j. The mathematics representation of the connectivity among the whole graph is the adjacency matrix $\mathbf{A} \in R^{N \times N}$, where N is the number of nodes, which equals the number of variables in the datasets. If $(v_i, v_j) \in \mathcal{E}$ then $a_{ij} \neq 0$, and if $(v_i, v_j) \notin \mathcal{E}$ then $a_{ij} = 0$. From the graph perspective, we describe the relations among nodes using the adjacency matrix $\mathbf{A}$. And the matrix will be learned and iterated through training.

Graph Learning The graph learning module uses input signals to generate an adjacency matrix to extract relations between variables. Unlike previous work [6] using pre-defined graphs to define the variable relationship with the physical distance of sensors, our module only relies on input data and does not require domain knowledge. As a result, such a self-learning graph will become a more common paradigm in graph neural network applications. The learned graph is generated in the following steps.

$$
\begin{aligned}
\mathbf{\Phi}_1 &= \tanh\left(\mathbf{W}_1 \mathbf{E}_1\right) \\
\mathbf{\Phi}_2 &= \tanh\left(\mathbf{W}_2 \mathbf{E}_2\right) \\
\mathbf{A} &= \mathrm{ReLU}\left(\tanh\left(\mathbf{\Phi}_1 \mathbf{\Phi}_2^T - \mathbf{\Phi}_2 \mathbf{\Phi}_1^T\right)\right) \\
\mathbf{A} &= \mathrm{topk}\left(\mathbf{A}\right)
\end{aligned}
\tag{1}
$$

where $\mathbf{E}_1$ and $\mathbf{E}_2$ represent two different variable embeddings, $\mathbf{W}_1$ and $\mathbf{W}_2$ are corresponding learnable model parameters, and $\mathbf{A}$ is the adjacency matrix. Two separate embeddings make the $\mathbf{A}$ asymmetrical, which can introduce more information. The topk$(\cdot)$ operation improves the sparsity of the adjacency matrix to help the graph convolution module focus on k nearest neighbors and reduce the calculation complexity in the following modules.

3.2 Graph Convolution Module

Given the input data with the corresponding mask matrix and the variable relationship graph from the graph learning module, our model merges the inputs $\mathbf{x}_t$ with their neighbors' information to generate an aggregated node representation $\mathbf{s}_t$ at time t. Specifically, the graph convolution module is constructed with D layers, which is formulated as

$$\begin{aligned} \mathbf{s}_t^{(0)} &= \mathcal{F}\left(\mathbf{x}_t \| \mathbf{m}_t \| \mathbf{h}_{t-1}\right) \\ \mathbf{s}_t^{(d)} &= \mathbf{A}\mathbf{s}_t^{(d-1)} \\ \mathbf{s}_t &= \mathcal{F}\left(\mathbf{s}_t^{(0)} \| \mathbf{s}_t^{(1)} \| \cdots \| \mathbf{s}_t^{(D)}\right) \end{aligned} \tag{2}$$

where $\mathbf{x}_t$ and $\mathbf{m}_t$ are input sequences and mask matrix at time t, $\mathbf{h}_{t-1}$ is the hidden state at time $t-1$, $\mathbf{A}$ is the graph adjacency matrix, and $\mathbf{s}_t^{(d)}$ is the aggregated node representation in layer d at time t. $\|$ is the concatenation operation. $\mathcal{F}(\cdot)$ is a feature fusion function implemented by a 1×1 convolution layer in our experiments.

3.3 Spatio-Temporal Fusion Module

The spatio-temporal fusion module receives the hidden state $\mathbf{h}_{t-1}$ from the previous time step and its aggregated nodes representation $\mathbf{s}_t$ at the current time step from the graph convolution module. Combining two information flows, this module generates current hidden state $\mathbf{h}_t$ at time t. Following previous work [13], we apply Gated Recurrent Unit (GRU) to control the proportion of information from previous time steps. The process of updating hidden states can be formulated as

$$\begin{aligned} r_t &= \sigma\left(\mathbf{W}_r(\mathbf{s}_t \| \mathbf{m}_t \| \mathbf{h}_{t-1}) + b_r\right) \\ u_t &= \sigma\left(\mathbf{W}_u(\mathbf{s}_t \| \mathbf{m}_t \| \mathbf{h}_{t-1}) + b_u\right) \\ \mathbf{c}_t &= \tanh\left(\mathbf{W}_c(\mathbf{s}_t \| \mathbf{m}_t \| r_t \odot \mathbf{h}_{t-1}) + b_c\right) \\ \mathbf{h}_t &= \mathbf{c}_t \odot u_t + \mathbf{h}_{t-1} \odot (1 - u_t) \end{aligned} \tag{3}$$

where r_t and u_t are reset and update gates, $\odot$ is element-wise multiplication. $\sigma(\cdot)$ and $\tanh(\cdot)$ are sigmoid and hyperbolic tangent activation functions. Thus, the hidden state $\mathbf{h}_t$ at time t can be updated and used for calculation at the next time step. After finishing all computation of T time steps, we fuse $\mathbf{s}_t$ and $\mathbf{h}_t$ to generate the final imputation of a branch.

3.4 Prediction Module

We introduce a bidirectional structure to combine forward and backward information. Compared to the unidirectional model, adding the backward branch can

utilize future information, making the imputed values more accurate. The final imputation $\widehat{\mathbf{Y}}$ is obtained by combining the outputs from forward and backward branches, which is formulated as

$$\begin{aligned} \hat{\mathbf{y}}_t &= \mathcal{F}\left(\mathbf{s}_t \| \mathbf{h}_{t-1}\right) \\ \widehat{\mathbf{Y}} &= \mathcal{F}\left(\mathrm{ReLU}\left(\widehat{\mathbf{Y}}^f \| \widehat{\mathbf{Y}}^b \| \mathbf{M}\right)\right) \end{aligned} \tag{4}$$

where $\hat{\mathbf{y}}_t$ is the reconstructed vector for $\mathbf{x}_t$ at time t, $\widehat{\mathbf{Y}}^f$, $\widehat{\mathbf{Y}}^b \in R^{N \times T}$ are imputed sequences from forward and backward branches separately, $\mathbf{M}$ is the mask matrix indicating the missing values location, and $\widehat{\mathbf{Y}} \in R^{N \times T}$ is the final imputation result. $\mathcal{F}(\cdot)$ is the feature fusion function, consistent with the Eq. (2), implemented by a 1×1 convolution layer in our experiment.

We define the loss for multivariate time series imputation as follows:

$$\mathcal{L}(\mathbf{Y}, \widehat{\mathbf{Y}}, \overline{\mathbf{M}}) = \sum_{n=1}^{N} \sum_{t=1}^{T} \frac{\langle \overline{m}_{n,t}, l(y_{n,t}, \widehat{y}_{n,t}) \rangle}{\langle \overline{m}_{n,t}, \overline{m}_{n,t} \rangle}, \tag{5}$$

where $\overline{\mathbf{M}}$ and $\overline{m}_{n,t}$ are logical binary complement of $\mathbf{M}$ and $m_{n,t}$; $\widehat{\mathbf{Y}}$ and $\widehat{y}_{n,t}$ are reconstructed data of missing values in $\mathbf{X}$; $\mathbf{Y}$ and $y_{n,t}$ are ground truth values at missing points in $\mathbf{X}$. $\langle \cdot, \cdot \rangle$ is the stand dot product. $l(\cdot, \cdot)$ is an element-wise error function, implemented by Mean Absolute Error (MAE) in our experiment.

4 Experiments

4.1 Datasets

We conducted experiments on four public time series datasets, which have various sizes and are representative of different application domains. The air quality datasets (AQI and AQI-36) [23, 26] are commonly used as a benchmark for time series imputation, which has high rates of missing values (about 26% in AQI and 13% in AQI-36). The traffic datasets (PEMS-BAY and METR-LA) [13] are originally used for time series forecasting tasks. To make them suitable for imputation tasks, we randomly masked 25% of the values in the traffic datasets to simulate missing values.

4.2 Baselines and Evaluation

We selected representative methods from three categories as baselines for our experiments: statistical methods (Mean, VAR), machine learning-based methods (kNN, MICE), and deep learning-based methods (GAIN, BRITS, and GRIN).

- **Mean**: Replace missing values with variable-level average.
- **kNN** [10]: Use k-nearest neighbor to impute missing values by averaging values of the $k = 10$ neighboring variables.

- **MICE** [1]: Multiple Imputation by Chained Equations setting a maximum number of iterations to 100 and the number of nearest features to 10.
- **VAR** [17]: Vector Autoregressive model with a one-step-ahead predictor.
- **GAIN** [24]: Generative Adversarial Imputation Nets with bidirectional recurrent encoder and decoder.
- **BRITS** [3]: Bidirectional Recurrent Imputation for Time Series, learning missing values in a recurrent dynamical system based on observed data.
- **GRIN** [6]: Graph Recurrent Imputation Network, using pre-defined graph and bidirectional 2-stage imputation.

To ensure a fair comparison, we used disjoint sequences to train and evaluate all the models, i.e., we trained the models with some sequences while testing them using other sequences for each dataset. For air quality datasets, we followed the prior work [23] and used 3^{rd}, 6^{th}, 9^{th} and 12^{th} months' data for testing and the rest for training. For traffic datasets, we followed [6] and split the data into three parts chronologically, using 70% for training, 10% for validation, and 20% for testing. We evaluate the models with three most commonly used metrics for time series forecasting and imputation tasks: Mean Absolute Error (MAE), Mean Square Error (MSE), and Mean Relative Error (MRE).

4.3 Results

Comparison with Baselines Our experimental comparison results (Table 1) show that our model outperforms all the compared models in all three metrics on the four datasets. In particular, for the AQI-36 dataset, our model improved the state-of-the-art method, GRIN, by a large margin, achieving a 30% decrease in MAE. In comparison, our model only achieved a slight improvement over the best-performing baseline, GRIN, on the traffic datasets. A possible reason is that the traffic datasets contain significantly more sensors that are geographically close to each other, making the sequences strongly correlated. As such, GRIN uses the geographic distances among sensors as domain knowledge to calculate the adjacency matrix to boost its performance. However, GRIN's excellent performance heavily relies on such prior knowledge and thus may not transfer to other datasets that have no such strong geospatial correlations.

Parameter Study We conducted parameter studies with respect to the number of neighbors k in Eq. (1) and the number of convolution layers D in Eq. (2). We selectively show some representative results (Figure 3), due to the limited space. The results on other datasets lead to similar conclusions. The parameter k controls the number of neighbors for each node, thus determining the density of the adjacency matrix in the graph learning module. Our experimental results on the parameter k (Figure 3a) shows the MAE remains relatively stable when $k \in \{2, 3, \cdots, 6\}$ but increases drastically when k goes under or beyond this range. It implies that an excessively small value of k causes the loss of important references from close neighbors for the imputation task, whereas a larger value of k (≥ 7)

Table 1. Performance comparisons on four real-world datasets. The best results are in boldface. The second-best results are underlined.

Datasets	Air Quality						Traffic					
	AQI-36			AQI			PEMS-BAY			METR-LA		
Methods	MAE	MSE	MRE(%)	MAE	MSE	MRE(%)	MAE	MSE	MRE(%)	MAE	MSE	MRE(%)
Mean	53.48	4578.08	76.77	39.60	3231.04	59.25	5.42	86.59	8.67	7.56	142.22	13.10
kNN	30.21	2892.31	43.36	34.10	3471.14	51.02	4.30	49.80	6.88	7.88	129.29	13.65
MICE	30.37	2594.06	43.59	26.98	1930.92	40.37	3.09	31.43	4.95	4.42	55.07	7.65
VAR	15.64	833.46	22.02	22.95	1402.84	33.99	1.30	6.52	2.07	2.69	21.10	4.66
GAIN	15.37	641.92	21.63	21.78	1274.93	32.26	1.88	10.37	3.01	2.83	20.03	4.91
BRITS	14.50	662.36	20.41	20.21	1157.89	29.94	1.47	7.94	2.36	2.34	16.46	4.05
GRIN	12.08	523.14	17.00	14.73	775.91	21.82	0.67	1.56	1.08	1.91	10.41	3.30
AGRN	**11.05**	**343.93**	**15.86**	**14.08**	**686.52**	**21.07**	**0.66**	**1.44**	**1.07**	**1.90**	**10.10**	**3.28**

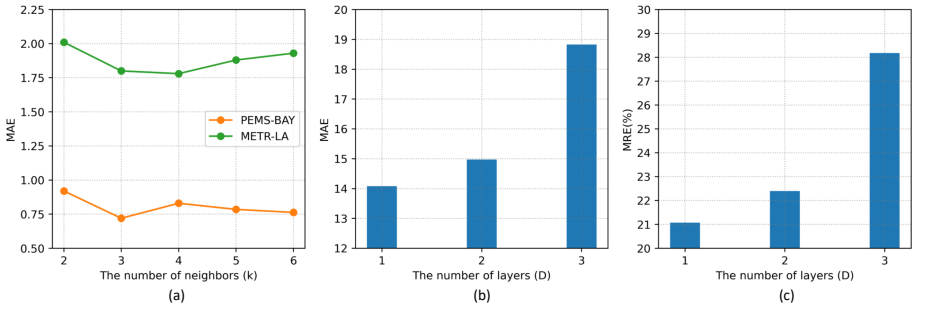

Fig. 3. Impact of parameters: (a) MAE under varying numbers of neighbors k on traffic datasets; (b) MAE and (c) MRE under varying numbers of convolution layers D on AQI dataset.

causes the model to consider irrelevant and distant neighbors, introducing extra noises and reducing the model's robustness.

The parameter D represents the number of layers used in the graph convolution module to aggregate representations of each node and its neighbors. Our experimental results on the parameter D (Figure 3b and Figure 3c) show our model's MAE and MRE consistently decrease as D increases on the AQI dataset. It aligns with our intuition that too many layers will cause over-smoothing and gradient vanishing issues with the graph convolution module, limiting the effectiveness of feature extraction in the subsequent spatio-temporal fusion module. We admit the above conclusion may not generalize to other datasets, as the optimal numbers of layers are dependent on the specific applications.

Ablation Study To test the impact of different modules on our model's performance reliably, we selected the AQI dataset, with the largest number of sensors among our experimental datasets, to conduct the ablation study. We compare our model with two variants of it: a) w/o graph: we remove the graph learning module and use the input $\mathbf{x}_t$ to replace the aggregated representation $\mathbf{s}_t$ in

Table 2. Ablation study on the AQI dataset.

AQI	MAE	MSE	MRE(%)
AGRN	**14.08**	**686.53**	**21.07**
w/o graph	19.70	1131.56	29.48
w/o bidirection	22.77	1365.25	34.06

Eq. (3); b) w/o bidirection: we remove the backward branch from the overall architecture and use the output of the forward branch $\widehat{\mathbf{Y}}^f$ as the final imputation. Our results (Table 2) show both modules contribute to the model's performance significantly, indicated by a notable increase in three metrics after removing either of them. Among the two modules, the overall bidirectional structural design plays a greater part in securing our model's superior performance, evidenced by a more drastic performance drop resulting from removing the backward branch.

5 Conclusion

In this paper, we propose a novel adaptive graph recurrent network (AGRN) to explore latent spatio-temporal dependencies for multivariate time series imputation. Instead of relying on pre-defined graphs, our graph learning module can generate an inter-variable graph adaptive to represent spatial dependencies, which improves our model's generality. Our extensive experiments demonstrate our model's superior performance to state-of-the-art baselines on several real-world datasets.

References

1. Azur, M.J., Stuart, E.A., Frangakis, C., Leaf, P.J.: Multiple imputation by chained equations: what is it and how does it work? International journal of methods in psychiatric research **20**(1), 40–49 (2011)
2. Beretta, L., Santaniello, A.: Nearest neighbor imputation algorithms: a critical evaluation. BMC medical informatics and decision making **16**(3), 197–208 (2016)
3. Cao, W., Wang, D., Li, J., Zhou, H., Li, L., Li, Y.: Brits: Bidirectional recurrent imputation for time series. Advances in neural information processing systems **31** (2018)
4. Che, Z., Purushotham, S., Cho, K., Sontag, D., Liu, Y.: Recurrent neural networks for multivariate time series with missing values. Scientific reports **8**(1), 1–12 (2018)
5. Cichocki, A., Phan, A.H.: Fast local algorithms for large scale nonnegative matrix and tensor factorizations. IEICE transactions on fundamentals of electronics, communications and computer sciences **92**(3), 708–721 (2009)
6. Cini, A., Marisca, I., Alippi, C.: Filling the g_ap_s: Multivariate time series imputation by graph neural networks. arXiv preprint arXiv:2108.00298 (2021)
7. Fang, C., Wang, C.: Time series data imputation: A survey on deep learning approaches. arXiv preprint arXiv:2011.11347 (2020)
8. Fortuin, V., Baranchuk, D., Rätsch, G., Mandt, S.: Gp-vae: Deep probabilistic time series imputation. In: International conference on artificial intelligence and statistics. pp. 1651–1661. PMLR (2020)

9. García-Laencina, P.J., Abreu, P.H., Abreu, M.H., Afonoso, N.: Missing data imputation on the 5-year survival prediction of breast cancer patients with unknown discrete values. Computers in biology and medicine **59**, 125–133 (2015)
10. Hastie, T., Tibshirani, R., Friedman, J.H., Friedman, J.H.: The elements of statistical learning: data mining, inference, and prediction, vol. 2. Springer (2009)
11. Jiang, W., Luo, J.: Graph neural network for traffic forecasting: A survey. Expert Systems with Applications p. 117921 (2022)
12. Kuppannagari, S.R., Fu, Y., Chueng, C.M., Prasanna, V.K.: Spatio-temporal missing data imputation for smart power grids. In: Proceedings of the Twelfth ACM International Conference on Future Energy Systems. pp. 458–465 (2021)
13. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:1707.01926 (2017)
14. Lu, W., Li, J., Wang, J., Qin, L.: A cnn-bilstm-am method for stock price prediction. Neural Computing and Applications **33**(10), 4741–4753 (2021)
15. Luo, Y., Cai, X., Zhang, Y., Xu, J., et al.: Multivariate time series imputation with generative adversarial networks. Advances in neural information processing systems **31** (2018)
16. Luo, Y., Zhang, Y., Cai, X., Yuan, X.: E2gan: End-to-end generative adversarial network for multivariate time series imputation. In: Proceedings of the 28th international joint conference on artificial intelligence. pp. 3094–3100. AAAI Press (2019)
17. Lütkepohl, H.: Vector autoregressive models. In: Handbook of Research Methods and Applications in Empirical Macroeconomics, pp. 139–164. Edward Elgar Publishing (2013)
18. Miao, X., Wu, Y., Wang, J., Gao, Y., Mao, X., Yin, J.: Generative semi-supervised learning for multivariate time series imputation. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 8983–8991 (2021)
19. Nelwamondo, F.V., Mohamed, S., Marwala, T.: Missing data: A comparison of neural network and expectation maximization techniques. Current Science pp. 1514–1521 (2007)
20. Pan, Z., Wang, Y., Wang, K., Chen, H., Yang, C., Gui, W.: Imputation of missing values in time series using an adaptive-learned median-filled deep autoencoder. IEEE Transactions on Cybernetics (2022)
21. Thomas, T., Rajabi, E.: A systematic review of machine learning-based missing value imputation techniques. Data Technologies and Applications **55**(4), 558–585 (2021)
22. Velicer, W.F., Colby, S.M.: A comparison of missing-data procedures for arima time-series analysis. Educational and Psychological Measurement **65**(4), 596–615 (2005)
23. Yi, X., Zheng, Y., Zhang, J., Li, T.: St-mvl: filling missing values in geo-sensory time series data. In: Proceedings of the 25th International Joint Conference on Artificial Intelligence (2016)
24. Yoon, J., Jordon, J., Schaar, M.: Gain: Missing data imputation using generative adversarial nets. In: International conference on machine learning. pp. 5689–5698. PMLR (2018)
25. Zhao, Y., Wang, Y.: Remaining useful life prediction for multi-sensor systems using a novel end-to-end deep-learning method. Measurement **182**, 109685 (2021)
26. Zheng, Y., Capra, L., Wolfson, O., Yang, H.: Urban computing: concepts, methodologies, and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **5**(3), 1–55 (2014)