GCRINT: Network Traffic Imputation Using Graph Convolutional Recurrent Neural Network

Van An Le*†, Tien Thanh Le‡, Phi Le Nguyen‡ Huynh Thi Thanh Binh‡, Rajendra Akerkar§, Yusheng Ji†*
*Department of Informatics, The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan
†National Institute of Informatics, Tokyo, Japan
‡Hanoi University of Science and Technology, Hanoi, Vietnam
§Western Norway Research Institute, Sogndal, Norway
Email: *†{anle, kei}@nii.ac.jp; ‡thanh.ltcb190211@sis.hust.edu.vn; ‡{lenp, binhht}@soict.hust.edu.vn; §rak@vestforsk.no

ICC 2021 - IEEE International Conference on Communications | 978-1-7281-7122-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICC42927.2021.9500687

***Abstract*—Missing values appear in most multivariate time series, especially in the monitored network traffic data due to high measurement cost and unavoidable loss. In the networking fields, missing data prevents advanced analysis and downgrades downstream applications such as traffic engineering and anomaly detection. Despite the great potential, existing imputation approaches based on tensor decomposition and deep learning techniques have shown limitations in addressing missing values of traffic data due to its dynamic behavior. In this paper, we propose Graph Convolutional Recurrent Neural Network for Imputing Network Traffic (GCRINT), a combination between Recurrent Neural Network (RNN) and Graph Convolutional Neural Network, for filling the missing values of network traffic data. We use a bidirectional Long Short-Term Memory network and Graph Neural Network to efficiently learn the spatial-temporal correlations in partially observed data. We conducted extensive experiments to evaluate our model by using two different datasets and various missing scenarios. The experiment results show that GCRINT achieves significantly low imputation errors and reduces the error by 35% compared to the state-of-the-art methods. GCRINT also helps to obtain a stable performance in the traffic engineering problem.**

I. Introduction

Recently, machine learning techniques have been applied to various networking problems, from simply learning or extracting knowledge to utilize and improve knowledge overtime. Building an effective machine learning model requires high accuracy in the historical measurement data. However, missing values in networking data are inevitable due to unexpected accidents or intended purposes (e.g., sparse monitoring [1]). This inappropriate data imposes a significant impact on the performance of downstream applications. A study in [2] showed that the accuracy of predicted traffic values is downgraded when increasing of missing ratio in network traffic data. Therefore, accurate recovering of missing values from the partially observed data plays an essential role in leveraging machine learning techniques for solving networking problems.

In the past decades, various approaches have been developed to address missing values in time series. The missing values can be filled by using statistical models such as Autoregressive Moving Average (ARMA) or Autoregressive Integrated Moving Average (ARIMA) [3]. However, these models are essentially linear and process time series independently. Therefore, they fail to exploit the correlation between different variables in multivariate time series. Matrix and tensor factorization were also applied to solve the data imputation problem. Many tensor completion algorithms have been proposed based on Alternating Least Square (ALS) such as Localized Tensor Decomposition (LTC) [4], gradient-based method such as Generalized Canonical Polyadic Tensor Decomposition (GCP) [5]. Among those, LTC [4] would be seen as the most recent work on estimating missing values, tailored for network traffic data. In LTC, the traffic matrix was represented as a 3-way tensor, including days, hours, and origin-destination dimension, to exploit the inherent relationship among higher-dimensional data. Also, LTC divides the huge 3-way traffic tensor into many sub-tensors with highly relevant data and performs Canonical Polyadic (CP) decomposition on these substructures. This approach is claimed to be more efficient than decomposing the original traffic tensor. However, all of the tensor completion algorithms mentioned above do not encapsulate the temporal correlation in time series data. In particular, the imputation data is formed by summing up the outer product of component matrices. The data imputed by CP in each time step cannot utilize the information from the previous and the following time steps. Moreover, most of the tensor completion methods rely on a strong assumption that the tensor data has a low-rank structure.

Other approaches for estimating the missing values are deep learning-based techniques. There are many imputation methods, which are based on Recurrent Neural Network (RNN), have been proposed. The RNN models, such as Long Short-term Memory (LSTM) or Gated Recurrent Unit (GRU), have achieved state-of-the-art results in many applications with time series and sequence data. Che, Purushotham, Cho, *et al.* proposed GRU-D [6], in which the missing data were represented as the combination of the last observed values and the mean value. GRU-D laid the foundation for other methods and demonstrated its significantly high performance on healthcare data with labels. Unfortunately, this method can not be directly applicable in an unsupervised manner on general datasets without the tag for each time series as our focus on network traffic data. Following GRU-D, Cao, Wang, Li, *et al.* introduced Bidirectional Recurrent Imputation for Time-

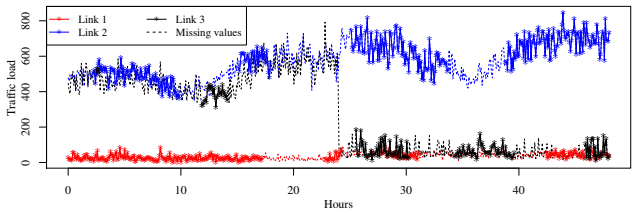


Fig. 1: Example of network link load traffic under block missing scenarios on two different days for three links in the Abilene network. The dot lines represent the missing blocks, while the stars indicate the observed data.

series (BRITS) as a novel bidirectional LSTM-based model for multivariate time series recovery [7]. A temporal decay factor and a linear layer were introduced in the BRITS model, which can help learn the spatial-temporal features of data recorded in irregular intervals. More recently, Generative Adversarial Network (GAN) has been used to impute missing values. Luo, Zhang, Cai, *et al.* proposed an End-to-end GAN (E^2GAN) [8] for extracting feature representation of time series and reconstructing it from the low-dimensional vector. E^2GAN model leverages the GRU-I cell, which was used in [7], to process the incomplete time series. However, E^2GAN only uses a unidirectional recurrent model and does not consider the correlation between the time series. Most recently, motivated by CP decomposition, NTC model was introduced in [1], which was specifically designed for imputing network traffic. However, by taking the index of each observed data as the input, NTC suffers from a scalability issue when the number of observed data in the training set is significantly large.

Although many efforts have been devoted to data imputation, most of the deep learning methods proposed so far have not focused on the network traffic data. Therefore, they suffer from the following critical problems. First, most of the methods mainly focused on the temporal relationship. Some approaches (e.g., BRITS [7]) have considered leveraging the spatial features, but they only cope with the static spatial correlations. Meanwhile, the correlations between the time series in the network traffic data vary significantly over time due to network behavior dynamics. Fig.1 visualizes the traffic load on three different links in the Abilene network to show the variation of the network behavior. Due to the dynamic of the routing scheme, the correlation between the links' traffic is inconsistent. Specifically, the traffic on link 3 shows a high correlation to that of link 2 on the first day, whereas, on the next day, it appears to be closely similar to link 1. Second, the existing models have not been evaluated by the network traffic dataset (e.g., the Abilene dataset). As the network traffic possesses unique characteristics, as mentioned above, the evaluation results for other data types are not likely to fit with network traffic data.

In this paper, we focus on network measurement data and propose Graph Convolutional Recurrent Neural Network for Imputing Network Traffic (GCRINT), a spatial-temporal deep learning model for network traffic imputation. GCRINT is a combination of the RNN-based model and Graph-based neural network. We design a multi-layers model whose each layer has two modules for learning features in time and space domains. Like BRITS, we use a bidirectional LSTM model for learning the temporal feature in the traffic data. To cope with the dynamic correlation mentioned above, we use Graph Convolutional Neural Network to exploit the spatial feature among the traffic flows automatically. Then, each layer's outputs are combined by a fully connected layer to obtain the final imputed data. Furthermore, to address the LSTM model's scalability in handling long sequences, the input is reduced by half after each layer. In this way, we can still learn the long-range dependency on deeper layers while lowering model complexity.

The contributions of this paper can be summarized as follows:

- We design the GCRINT model, which considers both spatial and temporal correlations to impute the missing network traffic data accurately.
- We conduct extensive experiments with various network traffic datasets to evaluate the prediction accuracy of the proposed method. We also study the performance of GCRINT when applying to other networking tasks such as traffic engineering.

This paper is organized as follows. The second section formulates the time series imputation problem and the overview of our solution. The proposed model and the mechanism to enhance the trainability of neural networks for data imputation are described in Section III. In Section IV, we describe how traffic engineering can leverage network traffic imputation. The next section presents experimental studies. Some conclusions are drawn in the final section.

II. Preliminaries

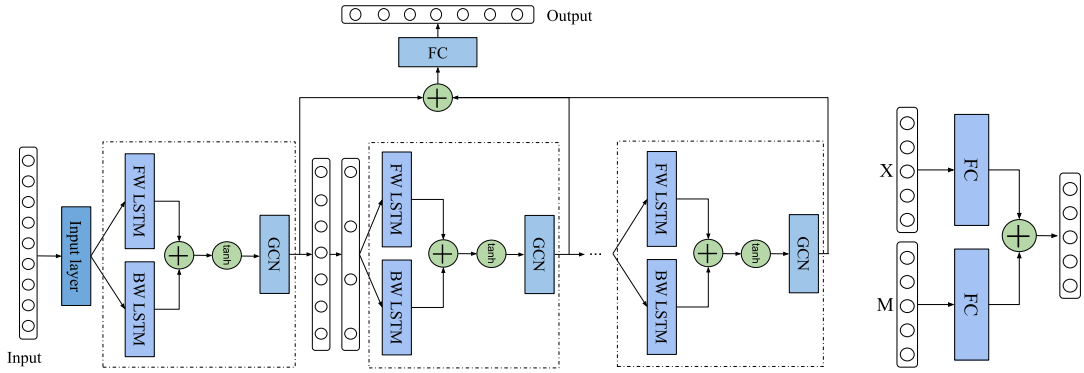
In this section, we first formulate the multivariate time series imputation problem and then present the background of Long Short-Term Memory and Graph Convolutional Neural Networks.

A. Problem description

We denote a partially observed data with D different time series (traffic flows), which are observed at N time steps, as $X \in \mathbb{R}^{N \times D}$. X is generally an incomplete matrix where each column $X^d = (x_1^d, ..., x_N^d)$ denotes the d-th time series and each row $X_t = (x_t^1, ..., x_t^D)$ is the data at time step t. We use a mask matrix $M \in \{0,1\}^{N \times D}$ to indicate the locations of the missing value, where $m_t^d = 1$ if x_t^d is observed, and 0, otherwise. The imputation problem can be formulated as:

$$\begin{aligned} &\min_{\hat{X}} \sum_{t=1}^{N} \sum_{d=1}^{D} |\hat{x}_t^d - x_t^d| \\ &s.t. \sum_{t=1}^{N} \sum_{d=1}^{D} m_t^d * |\hat{x}_t^d - x_t^d| = 0 \end{aligned} \tag{1}$$

where $\hat{x}_t^d \in \hat{X}$ is the imputed data of X.



(a) The overview of our proposed data imputation model (b) The input layer.

Fig. 2: The architecture of GCRINT model

B. Bidirectional Long Short Term-Memory Network

LSTM network is a special RNN that replaces the standard RNN units with the LSTM units. LSTM network has been proved to be stable and powerful for modeling long-range dependencies in various domains. Thus, it is well-suited for processing and making predictions based on time series or sequence data. Indeed, LSTM has been applied in many real-life sequence modeling problems. Bidirectional LSTM (BiLSTM) is a modification of the LSTM network where an additional backward LSTM is added to the model. The added backward network help to capture the information by the inverse order. Then, the outputs of the forward and backward networks are concatenated or averaged to obtain the final outputs. Therefore, the BiLSTM is widely used to solve the problems of estimating the missing data in a sequence, such as predicting the missing words in a sentence.

C. Graph Convolutional Neural Network

Graph Convolution Neural Network (GCN) is well-known for extracting the spatial features in data given its structural information in the form of a graph. Unlike the conventional Convolutional Neural Network (CNN), which is designed to exploit features of a fixed size, grid-based data structure such as images, GCN can learn the hidden representation of the unordered and variable-sized data structure [9]. In [10], the authors introduced the Diffusion Convolutional Neural Network (DCNN) to learn the spatial relations in the graphical data efficiently. Then, the DCNN has been applied in many studies [11], [12], to handle spatial-temporal data. In this study, we leverage the ability of DCNN to extract spatial features among traffic flows. The dynamic correlation of traffic flows can be learned from the data to improve imputation accuracy. In the next section, we will present in detail our proposed model GCRINT for imputing missing values in the network traffic data.

III. Graph Convolutional Recurrent Neural Network for Imputing Network Traffic

In this section, we present GCRINT (Fig.2a), which is the combination of BiLSTM and GCN for imputing network traffic. Overall, GCRINT has three main modules: an input layer, BiLSTM, and GCN layers. The input layer is responsible for extracting hidden features from the input, while the BiLSTM and GCN layers are for learning the temporal and spatial features in the data. GCRINT has multiple layers of BiLSTM and GCN; thus, it can handle temporal and spatial dependencies at different levels. After each layer, the number of time-steps in the sequence data is reduced by half. By skipping some time-steps in the sequence data, we reduce the computational complexity in the latter layers while still learning the long-term temporal information. We combine the outputs from each layer and use a fully connected layer to obtain the final output. Next, we present the details of each module in GCRINT.

A. Input layer

In contrast to the tensor completion-based approaches, which process the whole data at once, in the deep learning-based approaches, data is divided into sub-sequences and put into the imputation model sequentially. Let $[X_{1:T}, M_{1:T}]$ be the input of the GCRINT model in which $X_{1:T}$ is the traffic data of T time steps and $M_{1:T}$ is its corresponding mask matrix. Thus, the input of GCRINT is a 3D tensor $[T, D, 2]$ with two features: the traffic volume and the mask. In the input layer, we use two fully connected networks to obtain the feature representation of the input (as shown in Fig. 2b). After passing the input layer, we receive the output of $\mathcal{X}_{1:T}$, which is a 3D tensor with the size of $[T, D, H_I]$ (H_I is the number of hidden units of the fully connected network in the input layer). Then, $\mathcal{X}_{1:T}$ is fed into the BiLSTM layers.

B. Temporal feature learning with BiLSTM

Unlike the normal BiLSTM that receives the same input for both forward and backward LSTM networks, the BiLSTM layer in GCRINT takes inputs with different alignments for each LSTM propagation direction. Specifically, the forward LSTM takes $\mathcal{X}_{1:T-2}$ as input and obtains output $\mathcal{X}^f_{2:T-1}$ for time-steps from 2 to $T-1$. Similarly, $\mathcal{X}_{3:T}$ and $\mathcal{X}^b_{2:T-1}$ are the input and output of the backward LSTM. Therefore, from both LSTM networks, we obtain the output for the same time-steps $[2 : T-1]$. This technique was presented in [7] to overcome the backpropagation issue caused by the missing values in the

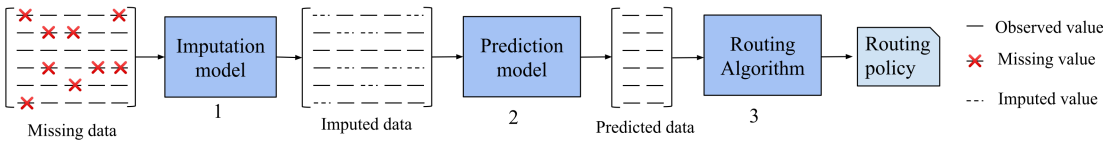


Fig. 3: Traffic Engineering leveraging network traffic imputation.

data. The final output of BiLSTM layer $\mathcal{X}_G$ is the mean of $\mathcal{X}^f_{2:T-1}$ and $\mathcal{X}^b_{2:T-1}$ (Eq. 2).

$$\mathcal{X}_G = \tanh\left((\mathcal{X}^f_{2:T-1} + \mathcal{X}^b_{2:T-1})/2\right) \tag{2}$$

The output of the BiLSTM layer is a 3D tensor with the size of $[T-2, D, H_B]$, where H_B is the hidden size of the LSTM networks.

C. Spatial feature learning with GCN

The GCN in each layer falls into the spatial-based, node-level graph neural networks. Each node in the graph represents the traffic of a network flow. We use Diffusion Convolutional Neural Network to extract the spatial relationships among the traffic flows, as shown in Equation (3).

$$Z = \sum_{k=0}^{K} f(P^k \mathcal{X}_G W_k) \tag{3}$$

where $\mathcal{X}_G$ is the input of GCN, K is the number of diffusion steps, P^k is the power series of the transition matrix, $f(.)$ is the activation function, and W_k is the learnable parameters. The transition matrix is computed by $P = D^{-1}A$ where A is the adjacency matrix of the graph, D is the diagonal matrix of node degrees, $D_{ii} = \sum_j A_{ij}$.

However, there is no explicit graph representing the relations among the traffic flows in this work. Therefore, we adopt the Self-adaptive Adjacency Matrix from [12] to learn the input data's adjacency matrix. The Self-adaptive Adjacency Matrix is obtained by Equation (4) proposed in [12].

$$A_{adp} = Softmax(ReLU(E_1 E_2^T)) \tag{4}$$

where E_1, E_2 are the learnable parameters. From (3) and (4), the DCNN with Self-adaptive Adjacency Matrix is represented as:

$$Z = \sum_{k=0}^{K} f(A^k_{adp} \mathcal{X}_G W_k) \tag{5}$$

The output of the GCN is a 3D tensor $Z_{2:T-1}$ with size $[T-2, D, H_G]$ (H_G is the hidden size of the GCN). Finally, The outputs of the GCN in each layer are combined and fed into a fully connected layer to obtain the final output.

IV. Downstream Applications

Although there are many proposed methods for imputing or recovering missing values in the network data [1], [4], most of them concentrated on improving the imputation error without considering the performance of downstream applications. In this section, we present how Traffic Engineering leverages network traffic imputation. The objective of the traffic engineering problem is to find a routing policy that minimizes the maximum link utilization (MLU) of the network and avoids traffic congestion. To this end, a common solution is to infer the future network traffic and then determine a routing policy that can adapt to the network traffic's dynamic. However, most traffic prediction methods proposed so far did not consider the impact of the missing values in the network data.

Figure 3 shows how traffic engineering benefits from the imputed network traffic data. First, the missing data is recovered using imputation models such as GCRINT, BRITS; then, a prediction model is trained using the imputed data. The prediction model predicts the future network traffic from the imputed data and uses it as input for calculating the routing policy. We use a simple LSTM model as the traffic prediction model and 2-Segment Routing [13] to calculate the routing policy. The performance of traffic engineering (MLU) is used to evaluate the performance of imputation methods.

V. Experimental Study

We design three types of experiments to evaluate our approach to network traffic imputation problem with real backbone network datasets. The experiment's results can be reproduced at [14].

A. Datasets and baseline methods

We conduct experiments on two real network datasets: Brain, and Abilene, available at [15]. Each dataset is divided into three sets: 70% for training, 10% for validating, and 20% for testing. Note that, although the Brain network has 161 nodes in total, most of them are regional nodes. Therefore, we only consider the aggregated traffic from 9 backbone nodes in the Brain network.

The baseline methods are:

- **GCP** [5]: the CP decomposition-based tensor completion approach.
- **NTC** [1]: the network traffic recovery model combines deep learning model (3D-CNN) and CP decomposition-based approach.
- **BRITS** [7]: This model bases on bidirectional recurrent network with GRU-I cell to impute time series. We do not compare with E^{2}GAN [8] because E^{2}GAN uses the same GRU-I cell, and their performances are relatively the same.

B. Performance metrics

We evaluate the imputation error using the Mean Absolute Error (MAE), which is calculated by Equation (6). For the traffic engineering problem, after obtaining the routing policy,

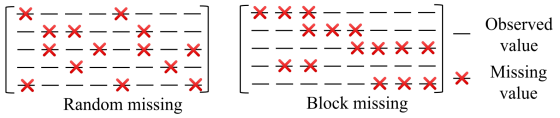
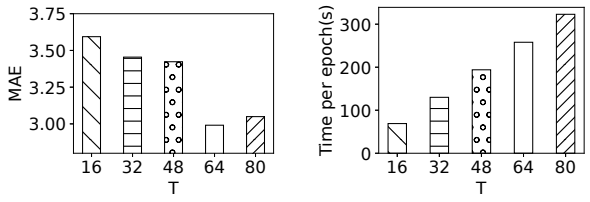


Fig. 4: Random and block missing schemes



(a) Imputation errors (b) Training time per epoch

Fig. 5: GCRINT on different input sizes

the MLU is calculated using the actual traffic matrix from the test set.

$$MAE = \frac{\sum_{t=1}^{N}\sum_{d=1}^{D}(1-m_t^d)|x_t^d - \hat{x}_t^d|}{\sum_{t=1}^{N}\sum_{d=1}^{D}(1-m_t^d)} \quad (6)$$

C. Generating synthetic missing data

We synthetically generate the missing data by removing $k\%$ (i.e., the missing rate) entries from the original data. The value of k is varied from 50 to 90. For each missing rate, the entries are removed by two patterns: *random missing* and *block missing* as shown in Fig.4. In the random missing, the entries are randomly removed, while in the block missing, a block of entries is removed consecutively.

D. Results

1) Impacts of the input size: The traffic data (consisting of N time-steps) is divided into $(N-T)$ sub-sequences by using a sliding window of size T. The missing data in every sub-sequence is then recovered. Finally, the imputed data is obtained by taking the average of the overlapped recovered sub-sequences. In this experiment, we study the impacts of T on the imputation errors of GCRINT. Figure 5a shows the imputation errors of GCRINT on Abilene dataset with 60% random missing values. As can be observed, the MAE decreases when T increases from 16 to 64 but increases beyond that. This phenomenon can be explained as follows. With a longer input sequence, the model may receive more information to recover the missing values, thereby increasing the imputation accuracy. Using $T = 64$, we can reduce 16.8%, 13.4% and 12.6% in MAE compared to the cases $T = 16, 32, 48$, respectively. However, when the sequence length is sufficiently large (i.e., $T = 80$), the imputation model becomes too complicated, which leads to the LSTM network's inherent drawbacks in handling long sequences.

The impacts of T on the training time is depicted in Fig.5b. As shown, there is a trade-off between the imputation accuracy and the computational complexity. The larger T, the more computational overhead in both model training and testing. Based on the experiment results obtained above, T should be set to a moderate value of 64. We use this value for all subsequent experiments.

2) Imputation accuracy: Figure 6 shows the performance comparison of all the methods in terms of MAE with different missing scenarios. Overall, the LSTM-based models (i.e., BRITS and GCRINT) outperform the CP decomposition-based approaches in the network traffic imputation problem. Our proposed model achieves the best performance in all the scenarios. All the methods have high imputation errors in the block missing in both datasets compared to the random missing scenario.

In comparison with GCP and NTC, GCRINT can reduce MAE by at most 80.5% and 68.7% on the Abilene dataset, and 72.0% and 54.4% on the Brain dataset. Comparing BRITS and GCRINT, while both models have almost the same MAE in the low missing rate (i.e., less than 70%), GCRINT achieves better performance when the missing rate increases. In the highest missing rate, GCRINT reduces MAE by about 35%.

3) Impacts of imputed data on traffic engineering: In this experiment, we do the traffic prediction and traffic engineering (Section IV) by using the imputed data of 90% missing rate. The optimal results are obtained by using the ground-truth data. Overall, traffic engineering results reflect the imputation methods' performance, as shown in Fig.7. The mean MLU of GCRINT is 22% lower than BRITS in the block missing scenario of the Abilene dataset. In the Brain dataset, GCRINT also reduces the MLU by about 70% to 80% on average, compared with that of GCP and NTC. Besides, the average MLU of our proposed model is only marginally over that of BRITS in the Brain dataset at about 3%.

Although the average MLUs of all the methods are close to the optimal, GCP and NTC suffer from a significantly high MLU in some absurd time-steps, especially in the Brain dataset. Therefore, the variances of GCP and NTC are considerably larger than those of GCRINT. Similarly, solving routing using imputed data by BRITS leads to high link utilization in many time-steps of the Abilene dataset (Fig.6a, 6b). In most of the cases, GCRINT achieves the lowest MLU.

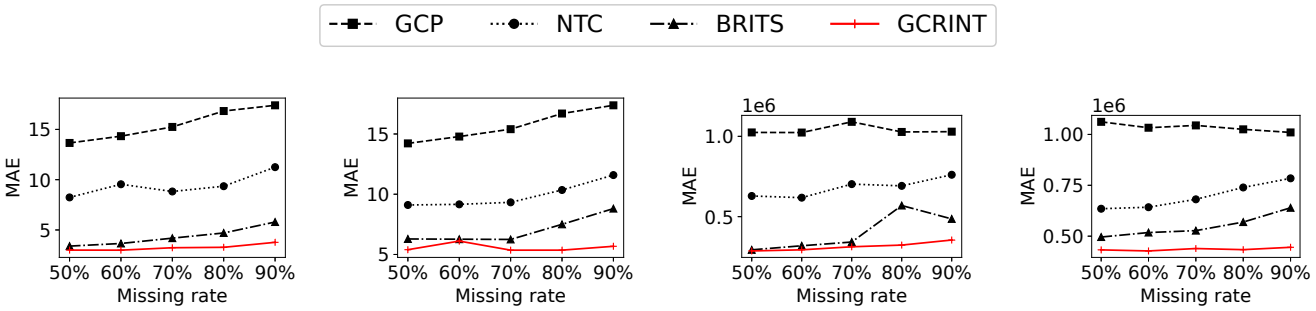
In conclusion, the recovered data provided by our proposed model facilitates the most stable performance in traffic engineering compared with all recently proposed methods.

VI. Conclusion

In this paper, we proposed a novel model, namely GCRINT, to address the network traffic imputation problem. To exploit the unique characteristic of traffic data, GCRINT is constructed with three main modules for extracting temporal and spatial features. Extensive experiments demonstrated our model's effectiveness in real network traffic datasets (e.g., Brain and Abilene datasets) with different missing scenarios. Moreover, we showed that GCRINT can improve the performance of downstream network applications such as traffic engineering.

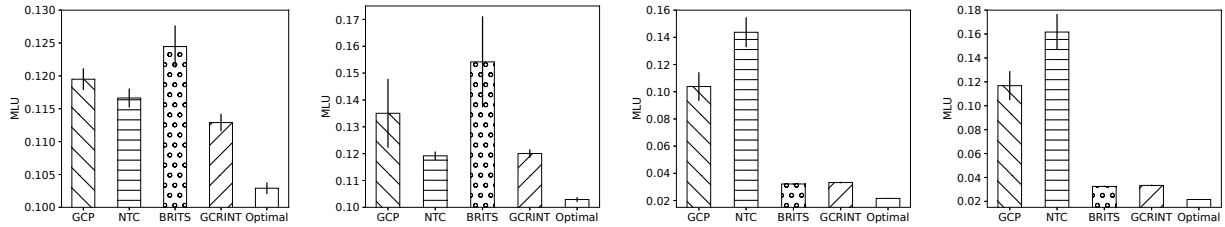
Acknowledgment

This work is supported in part by INTPART BDEM Grant No. 261685, and JSPS KAKENHI Grant No. JP20H00592, JP18KK0279.



(a) Abilene - random missing (b) Abilene - block missing (c) Brain - random missing (d) Brain - block missing

Fig. 6: The comparison of all methods in terms of MAE with different missing scenarios.



(a) Abilene - random missing (b) Abilene - block missing (c) Brain - random missing (d) Brain - block missing

Fig. 7: The comparison of imputation methods in traffic engineering problem.

REFERENCES

[1] K. Xie, H. Lu, X. Wang, G. Xie, Y. Ding, D. Xie, J. Wen, and D. Zhang, "Neural tensor completion for accurate network monitoring," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 1688–1697.

[2] V. A. Le, P. Le Nguyen, and Y. Ji, "Deep convolutional lstm network-based traffic matrix prediction with partial information," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 261–269.

[3] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[4] K. Xie, X. Wang, X. Wang, Y. Chen, G. Xie, Y. Ouyang, J. Wen, J. Cao, and D. Zhang, "Accurate recovery of missing network measurement data with localized tensor completion," *IEEE/ACM Trans. Networking*, vol. 27, no. 6, pp. 2222–2235, 2019.

[5] D. Hong, T. G. Kolda, and J. A. Duersch, "Generalized canonical polyadic tensor decomposition," *SIAM Review*, vol. 62, no. 1, pp. 133–163, 2020.

[6] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, vol. 8, no. 1, pp. 1–12, 2018.

[7] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "Brits: Bidirectional recurrent imputation for time series," in *Advances in Neural Information Processing Systems*, 2018, pp. 6775–6785.

[8] Y. Luo, Y. Zhang, X. Cai, and X. Yuan, "E 2 gan: End-to-end generative adversarial network for multivariate time series imputation," in *28th International Joint Conference on Artificial Intelligence*, 2019, pp. 3094–3100.

[9] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[10] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in neural information processing systems*, 2016, pp. 1993–2001.

[11] D. Andreoletti, S. Troia, F. Musumeci, S. Giordano, G. Maier, and M. Tornatore, "Network traffic prediction based on diffusion convolutional recurrent neural networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2019, pp. 246–251.

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

[13] R. Bhatia, F. Hao, M. Kodialam, and T. Lakshman, "Optimized network traffic engineering using segment routing," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, IEEE, 2015, pp. 657–665.

[14] https://github.com/vananle/GCRINT.git.

[15] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," English, in *Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium*, Apr. 2007.