TINet: Multi-dimensional Traffic Data Imputation via Transformer Network

Xiaozhuang Song, Yongchao Ye, and James J. Q. Yu*

Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation,
Department of Computer Science and Engineering,
Southern University of Science and Technology, Shenzhen 518055, China
11930376@mail.sustech.edu.cn

Abstract. Missing traffic data problem has a significant negative impact for data-driven applications in Intelligent Transportation Systems (ITS). However, existing models mainly focus on the imputation results under Missing Completely At Random (MCAR) task, and there is a considerable difference between MCAR with the situation encountered in real life. Furthermore, some existing state-of-the-art models can be vulnerable when dealing with other imputation tasks like block miss imputation. In this paper, we propose a novel deep learning model TINet for missing traffic data imputation problems. TINet uses the self-attention mechanism to dynamically adjust the weight for each entries in the input data. This architecture effectively avoids the limitation of the Fully Connected Network (FCN). Furthermore, TINet uses multi-dimensional embedding for representing data's spatial-temporal positional information, which alleviates the computation and memory requirements of attention-based model for multi-dimentional data. We evaluate TINet with other baselines on two real-world datasets. Different from the previous work that only employs MCAR for testing, our experiment also tested the performance of models on the Block Miss At Random (BMAR) tasks. The results show that TINet outperforms baseline imputation models for both MCAR and BMAR tasks with different missing rates.

Keywords: Data Mining · Attention Network · Data Imputation

1 Introduction

In recent years, models based on deep neural networks have been proposed and applied to solve traffic data-related problems, such as traffic prediction, traffic planning, and traffic simulation [19]. These deep learning methods are mostly built on sufficient and reliable historical traffic data, which is the fundamental basis of the modern Intelligent Transportation Systems (ITS). However, it

* This work is supported by the Stable Support Plan Program of Shenzhen Natural Science Fund No. 20200925155105002, by the General Program of Guangdong Basic and Applied Basic Research Foundation No. 2019A1515011032, and by the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001). James J.Q. Yu is the corresponding author.

is found that the traffic data collection process suffers from information loss or anomaly collection by various factors, and a large amount of data might be missing due to device failure in severe cases [1]. Rather than a high-quality dataset with available data, researchers obtain partially missing data or even sparse data most time. For instance, [15] found that nearly 50% of traffic data was missing in the past seven years at Alberta in Canada. Texas Transportation Institute (TTI) pointed out that their traffic management system had $16\% - -93\%$ incomplete data [12]. This makes deep learning models trained on ideal traffic data fragile in the practical application, as deep learning models may experience severe performance degradation when the data is corrupted [10]. Thus, building a robust and reliable traffic data imputation method is of great importance for the data-driven models [19].

In the past decades, a number of methods have been proposed to tackle the missing traffic data problem. Two of the most outstanding models among them are Bayesian Gaussian CANDECOMP/PARAFA factorization (BGCP) [2] and Stacked Autoencoder (SAE) [16]. BGCP considers the data imputation problem from the low-rank matrix approximation perspective. However, as it relies heavily on dimensional information, it meets a critical performance decline when the misses appear in random blocks or it needs to impute data with high missing rate. SAE learns to impute from the historical data, which makes it effective in solving the problems encountered by BGCP. However, the design of SAE's fully connected structure is not sufficiently flexible. Fixed-weight layers learn data blindly and require deeper network layers and more parameters to learn about the all the possible missing scenarios.

To tackle the problems listed above, in this work, we propose TINet, a novel missing traffic data imputation framework based on the Transformer structure [13]. The main contribution of our work can be listed as follows:

1. To the best of our knowledge, this is the first work that adopts a Transformer-based deep learning framework to study the problem of missing traffic data imputation. The results show that our model outperforms other baselines.
2. We propose a multi-dimensional embedding representation for the discrete attributes for traffic data, with the spatial embedding learned from a random walk-based graph embedding and the temporal embedding learned from embedding layers.
3. We conduct comprehensive experiments on two real-world datasets to compare TINet with baseline models and give a detailed analysis of experimental results.

2 Related Work

Missing traffic data imputation methods have been researched for decades. The mainstream of traffic data imputation methods can be divided into two categories: (1) Tensor completion models, (2) Data-driven models. Tensor completion models is proposed base on the low-rank property of traffic data, and models built based on the CANDECOMP/PARAFA (CP) decomposition method is among the most representative models for its imputation performance. [3] introduces the idea of tensor completion into traffic data with a Bayesian Gaussian CANDECOMP/PARAFA (BGCP) model, while [4] proposes a Bayesian Probabilistic Matrix Factorization (BPMF) method for traffic data imputation and traffic forecasting. Both these two models perform well under the Missing Completely At Random (MCAR) imputation task. However, it shows a significant performance drop when handling the block missing task. We will discuss this in detail in Section 5.

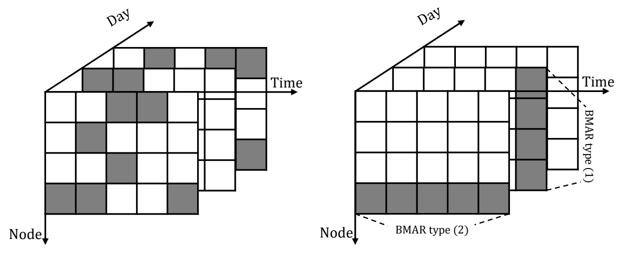
Data-driven approaches largely tackle the above problems for their scalable learning process and use more parameters to capture the data correlation. Stacked Autoencoder (SAE) [16] builds a deep autoencoder structure with a fully connected Network. GAIN [17] uses a generative adversarial training style to generate missing data with the available data. However, GAIN has poor generalizing ability as it trains and tests its model on the same dataset, which causes a severe overfitting problem. Transformer [13] is the current state-of-the-art model for dealing with sequence data in the field of natural language processing. Transformer and its conceptual progeny have topped many benchmark leaderboards in sequence data learning tasks [13]. A Transformer network uses self-attention mechanism to dynamically adjust the weight of input data, and it uses residual connections, layer normalization for better performance [13]. One of the most outstanding Transformer-based work is BERT [6]. It pre-trains the model with massive existing data and two subtasks, Masked Language Model and Next Sentence Prediction, which requires the model to predict the words it masks and the next sentence, respectively. The success of Transformer, especially BERT, shows the great potential of self-attention-based structure in sequence data. In this work, we will introduce how to apply the Transformer structure network to traffic data.

3 Preliminaries

In this section, we elaborate on the preliminaries to the missing traffic data problems. The problem formulation and introductions to the investigated missing types are presented as follows.

3.1 Problem formulation

In this work, we treat traffic data as a sequence data $D = \{x_1, x_2, \cdots, x_n\}$, with each value in the sequence representing its traffic speed value. Two adjacent data in the sequence have a fixed interval of s minutes, and t_i represents the time of x_i. Each sequence has a corresponding day of week wd_D and a corresponding traffic node v_j. We denote the sequence with missing data with $\widehat{D}$, and value in D, $\widehat{D}$ with x_i, $\widehat{x}_i$, respectively. To simulate the missing data, we introduce a mask tensor $M = \{m_1, m_2, \cdots, m_n\}$ with its value $m_i \in \{0, 1\}$. Therefore, we can get the sequence $\widehat{D}$ with $\widehat{x}_i = x_i \times m_i$. Given D and $\widehat{D}$, we aim to find a



(a) MCAR expample. (b) BMAR example.

Fig. 1. Visualization example for different missing types. Gray cells refer to missing values, while the white ones are available. For MCAR, missing values appear randomly. For BMAR, missing values appear in block as introduced in Section 3.2

model f to learn the objective:

$$\min_{\theta} \sum_{i} (x_i - f(\widehat{x}_i|\theta))^2 \tag{1}$$

where θ is the parameters of f.

3.2 Missing Types

Missing data have different types. In our study, we mainly study the following two missing types: **MCAR** : The data is missing completely at random; **BMAR** : The data is missing completely at random and appears as blocks. In our experiments, the block missing appears randomly as one of the following types: (1) One or multiple nodes lost their data at arbitrary time. (2) All day data are lost for particular nodes. The visualization examples of MCAR and BMAR can be seen from Figure 1.

4 TINet

In this section we describe the architecture of our proposed model TINet. Figure 2 depicts the overall architecture.

4.1 Model Architecture

TINet is mainly composed of Transformer modules, where a Transformer module comprises two sub-layers: a Multi-Head Attention sublayer and a position-wise Fully Connected sublayer. A residual connection is incorporated around each of the two-sublayers with layer normalization. According to [13], to promote the Transformer-based model's performance, we set all Transformer layers to produce outputs of dimension d_o. The final Transformer module's output feeds to a fully connected network with size $[d_o, 1]$. Then we can get the final output imputation $\widehat{D}$.

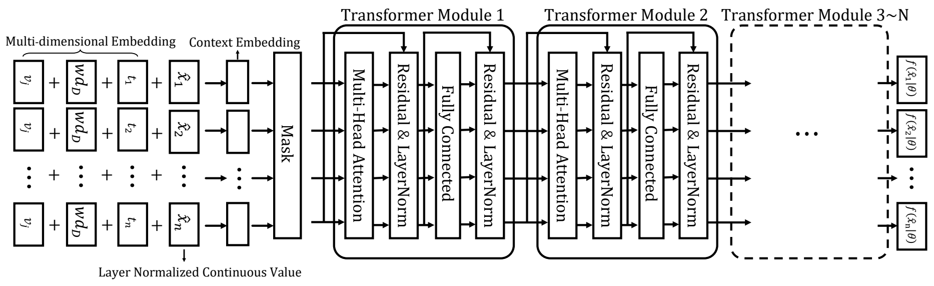


Fig. 2. Overview of TINet.

4.2 Context Embedding

TINet computes a context embedding from its normalized continuous value and discrete attributes for each input value x_i.

Layer Normalized Continuous Value: For continuous value x_i, we normalize it with z-score layer normalization:

$$\bar{x} = \frac{x - \mu}{\sigma} \tag{2}$$

where μ and σ are the mean and variance of corresponding layer input.

Multi-dimensional Embedding: For a point x_i in the traffic sequence D, TINet selects the day of week wd_D, its daytime t_i, node id v_j as its discrete attributes. For wd_D and t_j, we use two projection layers with l_1 and l_2 dimensions, respectively. For node id v_j, we compute its embedding by applying the Deep Walk method [9]: Firstly, node sequences are generated based on random walk. The transition probability of random walk is defined as

$$P(v_j \mid v_i) = \begin{cases} \frac{W_{ij}}{\sum_{j \in \mathcal{N}(v_i)} W_{ij}}, & v_j \in \mathcal{N}(v_i) \\ 0, & otherwise \end{cases} \tag{3}$$

where $\mathcal{N}(v_i)$ represents the neighbour nodes of v_i, e_{ij} represents the edge from v_i to v_j, W_{ij} is the historical travel data statistics from v_i to v_j[1], respectively. After finishing the random walk, we apply Skip-Gram algorithm [8] to learn the embedding of node. Assume a walking sequence $Walk = \{v'_1, v'_2, \cdots, v'_n\}$, the learning process of Deep Walk can be summarized by the following formula:

$$\max_{\vartheta} \log \mathrm{P}\left(\left\{(v'_{i-w}), \cdots, (v'_{i+w})\right\} \mid \vartheta(v'_i)\right) \tag{4}$$

For a given node v'_i in the walking sequence $Walk$, $(v'_i) \in \mathbb{R}^N$ represents its one-hot encoding vector. We feed (v'_i) to a network ϑ. The objective is to maximize

[1] For PeMS dataset which will be introduced in Section 5, we represent W_{ij} by the distance between v_i and v_j as there is no historical travel data statistics

the output probability of the nodes near v_i' in $Walk$ with window size w. The intuition of this idea is that the nodes with a closer distance in the walking sequence have similar representation on latent space [9]. ϑ commonly adopts a two-layer fully connected encoder-decoder structure. The first and second fully connected layers are of size $[N, L_v]$ and $[L_v, N]$, where N is the number of node and L_v is hidden size. Typically, the first layer is used to extract embedding vectors of input nodes, the second layer is used to transform them into nodes' one-hot encoding vector close in the walk sequences.

After the embedding of wd_D, t_i and v_j is learned, we concatenate them with normalized continuous value as the context embedding and feed them into the first Transformer module.

4.3 Multi-head Attention Computaion

The context embedding is firstly associated with three learnable weight parameters W_Q, W_K, and W_V, each with size $[L_{em}, k]$ where L_{em} is the length of context embedding, and k is the latent dimension. This process is reflected by

$$\begin{aligned} \mathrm{Q} &= \mathrm{X} \cdot W_{\mathrm{Q}} \\ \mathrm{K} &= \mathrm{X} \cdot W_{\mathrm{K}} \\ \mathrm{V} &= \mathrm{X} \cdot W_{\mathrm{V}} \end{aligned} \tag{5}$$

where Q, K, V represents the *query* vector, *key* vector and *value* vector in the self-attention mechanism [13]. $\{\cdot\}$ represents the matrix multiplication operator. We can see that the self-attention mechanism builds the dynamic weights between input data with learnable parameters, and a detailed explanation for attention can refer to the previous work [14]. The final layer output of self-attention layer is:

$$\mathrm{Att}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = \mathrm{softmax}(\frac{\mathrm{Q}\mathrm{K}^{\mathrm{T}}}{\sqrt{d_k}})\mathrm{V} \tag{6}$$

where the resulting output is a context vector of dimension k. $\sqrt{d_k}$ represents the square root of *key* vector, and it leads to more stable gradients. Furthermore, we use a n-head attention mechanism [13], which aims to mine n different attention scores by setting n W_{Q}, W_{K} and W_{V}. [5] shows that a proper number of attention heads can better mine the information in data and enhance the model performance.

5 Case Studies

In this section, we introduce the settings of our experiment. We conduct experiments on two real-world datasets and compare TINet with other baseline models.

5.1 Experimental Settings

Experimental Environment All experiments are performed on a Linux Server (CPU: Intel(R) Xeon(R) CPU E5-2620v4, GPU: GeForce RTX2080Ti, System: Ubuntu 18.04). All models are conducted with Python 3.7.

Hyperparameter Settings The number of Transformer modules is 4, and the attention heads are 2. The dimensions l_1, l_2 of projection layers are both set to 8. The self-attention layer's hidden size is set to 16, while the hidden size of the Fully Connected sublayer and the final output dimension d_o are both set to 32. The hidden size L_{em} of Deep Walk is set to 16. The proportions for the training set, validation set, and test set are 70%, 10%, and 20%, respectively. For both TINet and baseline models, we run 20 times and take the results' average as the final comparison. We use Adam optimizer [7] to optimize model parameters. During training, the learning rate is 10^{-3}, and the batch size used in training is 16. We choose the max training epoch 500 for all data-driven models. All the parameters above are set from a empirical grid-search finetuning results. Data-driven models all use an early stopping strategy to avoid the overfitting problem where the validation loss no longer decreases for three consecutive epochs.

Dataset Description We conduct our experiments on two open datasets:

1. PeMS Dataset[2]: This dataset collects real-time traffic data from nearly 40,000 individual detectors across all major districts in California. Followed by previous work [2,16], we choose the District 5 data ranging from January 1, 2013 to December 31, 2013. All the collected data is preprocessed into 144 road sections, and their information is aggregated every five minutes, which means the data of one day is embedded in a matrix of 288×144.
2. Shenzhen Dataset[3]: This dataset collects raw GPS trajectories in Shenzhen from October 8, 2019 to October 14, 2019. We select the area centered on Futian District area as it only has few missing data. Therefore, we can make intentional data missing simulation expediently. This dataset involves 802 areas. We use the map-matching algorithm to process the raw data and aggregate their traffic information every 5 minutes.

For missing rate selection, we consider the following two aspects: (1) The performance gap of the models under low missing rate is not significant. (2) The design of TINet method is more inclined to address the imputation tasks with high missing rates. Since that, the missing rates selected in our experiment is 0.1, 0.5, 0.8, 0.9. Before training, we generate datasets with different missing rates. For MCAR, we mask the value in dataset randomly. For BMAR, we randomly select one of the two BMAR types as we introduce in Section 3.2. For instance, if type (1) of BMAR is selected, we randomly select an arbitrary time in a day, and mask all data at that time with a high probability of 95%[4]. We repeat this process until the proportion of missing data in the dataset meets the requirements.

[2] https://dot.ca.gov/programs/traffic-operations/mpr/pems-source

[3] https://opendata.sz.gov.cn/data/dataSet/toDataDetails/29200_00403602

[4] We don't set this to be a 100% probability, which somehow rarely happens in practice.

5.2 Baselines and Evaluation Metrics

We compare TINet with two categories of methods: matrix completion and data-driven methods. In the former class, we select BPMF [4] and BGCP [2] as the baselines. On the other hand, for data-driven models, we adopt HA [12], SAE [16], and GAIN [17] as the baselines. Besides, we introduce TINet$_w$, which is a variant of TINet without multi-dimensional embedding, to evaluate the effectiveness of such embedding. We also choose three performance metrics to compute the difference between ground truth Y_i and masked data $\widehat{Y}_i$, including:

1. Mean Absolute Error (MAE):

$$\mathrm{MAE} = \frac{1}{m}\sum_{i=1}^{m} \|(Y_i - \widehat{Y}_i)\| \tag{7}$$

2. Root Mean Squared Error (RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(Y_i - \widehat{Y}_i)^2} \tag{8}$$

3. Mean absolute percentage error (MAPE):

$$\mathrm{MAPE} = \frac{\|Y - \widehat{Y}\|}{\|Y\|} \times 100\% \tag{9}$$

MAE and RMSE measures the average prediction error of the model. MAPE measures the percentage gap between predicted value and target value.

5.3 Experimental Results

Table 1 shows the experimental results, which corresponds to the model's performance for imputing two missing data types on the two datasets. Notably, for GAIN, we can see a significant error in our experiment. We attribute this performance to GAIN's training strategy. GAIN trains and tests its imputation ability on the data without splitting into training, validation, and testing sets. However, we consider this not meeting the practical scenarios. In our experiment, we train models on separate training and validation sets. Under this experimental settings, the obtained results of GAIN are not satisfactory. Furthermore, we can see that TINet$_w$ has an apparent performance degradation compared to TINet, which shows the effectiveness of multidimensional embedding. In the rest of this section, we discuss the experiment results from multiple perspectives:

Matrix Completion Models v.s. Data-driven Models From Table 1 we can see that the data-driven models have a robust performance under high missing rate scenarios. Take the performance of MCAR task on PeMS dataset with 0.9 missing rate as an example, the MAPE of TINet and SAE is 3.05% and 3.63%, while BGCP and BPMF's MAPE reach 14.12% and 15.38%, which are

Table 1. Experimental Results for MCAR task

Performance on MCAR task for PeMS dataset (MAE/RMSE/MAPE(%))				
Missing rate	0.1	0.5	0.8	0.9
HA	3.87/4.29/7.70	3.89/4.88/8.03	3.60/4.26/7.52	3.86/4.84/8.01
BPMF	1.88/2.88/3.60	3.21/5.38/6.86	5.00/7.47/10.31	7.78/10.41/15.38
BGCP	**1.34/1.78/2.55**	1.86/2.49/3.11	4.05/5.21/7.77	7.38/9.96/14.12
SAE	2.17/2.86/3.75	2.03/2.70/3.52	2.17/2.94/3.74	1.92/2.81/3.63
GAIN	38.62/42.42/76.20	40.42/43.50/79.03	42.01/44.25/82.06	46.78/47.81/91.31
TINet$_w$	1.84/2.60/3.32	1.57/1.63/2.54	1.84/2.50/3.35	1.85/2.64/3.38
TINet	1.69/2.41/2.90	**0.97/1.33/1.64**	**1.76/2.34/3.01**	**1.76/2.42/3.05**
Performance on MCAR task for Shenzhen dataset (MAE/RMSE/MAPE(%))				
Missing rate	0.1	0.5	0.8	0.9
HA	10.65/11.93/26.77	10.71/12.74/27.44	10.69/12.97/26.81	10.50/12.81/26.83
BPMF	3.94/5.67/9.43	4.66/6.59/10.79	5.49/7.67/12.91	6.61/8.76/15.61
BGCP	3.51/4.56/8.27	4.25/5.36/10.46	4.89/5.87/11.54	5.73/6.92/13.50
SAE	2.56/3.37/6.02	2.23/2.98/5.25	2.38/2.75/5.60	2.63/3.47/6.19
GAIN	30.04/33.54/77.48	32.54/35.70/81.99	35.11/37.21/88.09	36.08/37.98/90.10
TINet$_w$	1.44/1.61/3.27	1.96/2.33/4.61	2.24/2.70/5.27	2.55/3.14/6.00
TINet	**1.07/1.33/2.52**	**1.26/1.41/3.36**	**1.73/2.11/4.57**	**2.15/2.47/5.20**

Table 2. Experimental Results for BMAR task

Performance on BMAR task for PeMS dataset (MAE/RMSE/MAPE(%))				
Missing rate	0.1	0.5	0.8	0.9
HA	4.13/4.55/8.85	4.55/5.42/8.56	4.28/5.10/8.32	4.22/5.43/8.21
BPMF	10.74/13.69/23.92	13.93/14.40/26.37	19.40/29.42/34.81	19.65/20.01/37.41
BGCP	12.38/14.87/24.50	15.97/20.56/37.18	21.40/27.68/42.89	28.72/36.64/53.11
SAE	2.20/2.90/3.78	4.95/6.69/9.60	5.32/7.04/10.58	5.54/7.25/10.70
GAIN	42.64/45.78/83.08	41.85/44.86/82.21	44.22/45.42/86.60	41.93/45.05/81.96
TINet$_w$	2.48/3.46/4.09	3.13/4.68/4.83	4.43/6.13/8.63	5.05/6.43/10.46
TINet	**1.94/2.46/2.95**	**1.64/2.72/2.98**	**1.82/2.44/3.03**	**1.77/2.48/3.05**
Performance on BMAR task for Shenzhen dataset (MAE/RMSE/MAPE(%))				
Missing rate	0.1	0.5	0.8	0.9
HA	11.24/12.74/28.19	10.94/12.39/27.73	10.91/12.25/27.37	10.82/12.23/27.45
BPMF	8.56/10.62/21.68	9.87/12.47/24.62	8.91/11.37/22.36	20.19/26.10/51.38
BGCP	12.50/16.98/32.08	13.25/17.98/35.23	18.62/24.38/45.64	21.20/27.33/55.30
SAE	3.06/3.51/7.32	5.10/6.65/12.74	5.28/6.81/13.06	5.46/7.09/13.39
GAIN	36.97/39.72/91.21	32.13/35.52/79.87	34.18/36.95/86.23	35.40/37.26/88.43
TINet$_w$	1.47/1.83/3.46	4.07/5.51/11.16	4.83/6.94/12.70	5.40/7.16/13.21
TINet	**1.12/1.35/2.59**	**2.78/4.19/7.41**	**3.43/4.87/9.45**	**3.99/5.72/10.47**

nearly four times higher than the former two. The high missing rate makes the numerical information of each dimension of the matrix or tensor largely lost. Nonetheless, the matrix and tensor factorization techniques rely on this information for inferring the posterior numerical distribution of values on specified positions. Therefore, a high missing rate brings a significant performance drop on BGCP and BPMF. Besides, the performance of data-driven models is relatively stable under both tasks. We attribute this stability to the scalable parameter learning scheme for deep learning models, as they can continuously learn new data and update network parameters. However, BGCP and BPMF methods both perform low rank approximation to the structure of the input data itself, and do not have such scalability. Meanwhile, BGCP and BPMF are underperforming on BMAR. This can also be attributed to its dependency on dimensional numerical information, and the blocks of missing data can hardly provide such information. There is another interesting observation in the comparison, i.e., BGCP and BPMF do not perform well for MCAR task on Shenzhen dataset. We can see the MAPE of BGCP and BPMF of MCAR task on Shenzhen dataset with 0.1 missing reach 8.27% and 9.43%, respectively. CP decomposition enforces a strict structure assumption by modeling hidden parameters for each dimension, which makes it suitable for highly structured data. However, this also undermines the generalization capability of CP decomposition method for the scenarios that the correlation among data is relatively trivial [11].

Comparison between TINet and SAE From Table 1 we can obtain a direct comparison among data-driven models. As GAIN performs poorly during our test due to the overfitting problem, we mainly discuss the difference between SAE and TINet in this comparison. It can be seen that TINet has a slight performance advantage over SAE in all tasks. Moreover, TINet's performance advantage on the Shenzhen dataset is even more significant. In particular, under 0.1 missing rate, TINet got 2.52% MAPE for MCAR task and 2.59% MAPE for BMAR task, while SAE's MAPE reach 6.02% and 7.32%, respectively. We attribute this to the difference in the model architecture. As SAE mainly uses a naïve, fully connected layer as the basic module, this fixed-weight connection makes it difficult for the model to impute data with misses appear in random positions. Furthermore, it needs a larger amount of parameters and a deeper network to learn the parameters suitable for all missing cases for this fixed weight structure. However, this is somehow difficult and computationally expensive. On the other hand, The self-attention mechanism applied in TINet can dynamically adjusts the weights of networks according to the input data, making the model more robust and effective.

Impact of Missing Types The main difference between MCAR and BMAR is that BMAR causes a complete loss of dimensional information, impacting both BPMF and BGCP. Besides, data-driven models also suffer from a slight performance drop. TINet has the smallest performance gap between the two tasks. We attribute this robust performance to the self-attention mechanism and the multi-dimensional embedding used in TINet, as it models the spatial-temporal relationship among traffic data in a more reasonable way. Additionally, TINet without a self-attention layer is a special case of SAE, and their difference in metrics can reveal the effectiveness of the self-attention layer directly.

5.4 Limitations of Applying Self-attention Mechanism on Long Sequence Data

Applying self-attention mechanism on traffic data has computational and memory requirements which are quadratic with the input sequence length. With contemporary computing hardware and model sizes, this typically limits the input sequence [18]. However, this does not meet the needs of traffic sequence data, as the multi-dimensional traffic data requires a huge attention matrix, which is impractical. Thus, a natural question arises: can we achieve the empirical performance of quadratic full self-attention performance with a network with fewer parameters? In this work, TINet computes time embedding and graph embedding of node to avoid calculating attention of multi-dimensional data. This method is simple and practical to improve the performance of the model. Nevertheless, how to effectively solve the calculation and memory requirements of the self-attention mechanism on long-sequence multi-dimensional traffic data is still a problem worthy of research.

6 Conclusions

In this paper, we study the multi-dimensional traffic data imputation problem with two missing patterns. We design TINet, an effective Transformer-based model to impute data with random data missing and block data missing scenarios. The self-attention-based Transformer Module makes TINet circumventing the limitations of Fully Connected Network. Furthermore, the multi-dimensional embedding not also improves the performance of TINet, but also avoids the excessive calculation of attention on multi-dimensional data. We evaluate TINet and compare the results with other baselines. The result shows that TINet develops superior imputation performance under most scenarios. We also discuss the limitations of applying self-attention mechanism on long sequence traffic data. To overcome these existing drawbacks, we will further extend current work into designing effective and efficient Transformer models for multi-dimensional traffic data in the future.

References

1. Chen, H., Grant-Muller, S., Mussone, L., Montgomery, F.: A study of hybrid neural network approaches and the effects of missing data on traffic forecasting. Neural Computing & Applications **10**(3), 277–286 (2001)
2. Chen, X., He, Z., Sun, L.: A bayesian tensor decomposition approach for spatiotemporal traffic data imputation. Transportation research part C: emerging technologies **98**, 73–84 (2019)

3. Chen, X., He, Z., Sun, L.: A bayesian tensor decomposition approach for spatiotemporal traffic data imputation. Transportation Research Part C: Emerging Technologies **98**, 73–84 (2019)
4. Chen, X., Sun, L.: Bayesian temporal factorization for multidimensional time series prediction. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)
5. Correia, G.M., Niculae, V., Martins, A.F.: Adaptively sparse transformers. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing. pp. 2174–2184 (2019)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186 (2019)
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations, San Diego, CA, USA, May 7-9 (2015)
8. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
9. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 701–710 (2014)
10. Redman, T.C.: If your data is bad, your machine learning tools are useless. Harvard Business Review **2** (2018)
11. Sidiropoulos, N.D., De Lathauwer, L., Fu, X., Huang, K., Papalexakis, E.E., Faloutsos, C.: Tensor decomposition for signal processing and machine learning. IEEE Transactions on Signal Processing **65**(13), 3551–3582 (2017)
12. Smith, B.L., Scherer, W.T., Conklin, J.H.: Exploring imputation techniques for missing data in transportation management systems. Transportation Research Record **1836**(1), 132–142 (2003)
13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. arXiv preprint arXiv:1706.03762 (2017)
14. Vig, J., Belinkov, Y.: Analyzing the structure of attention in a transformer language model. In: Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. pp. 63–76 (2019)
15. Xu, J.r., Li, X.y., Shi, H.j.: Short-term traffic flow forecasting model under missing data. Journal of Computer Applications **30**(4), 1117–1120 (2010)
16. Yanjie Duan, Yisheng Lv, Wenwen Kang, Yifei Zhao: A deep learning based approach for traffic data imputation. In: IEEE Conference on Intelligent Transportation Systems. pp. 912–917 (2014)
17. 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)
18. Zaheer, M., Guruganesh, G., Dubey, K.A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., Ahmed, A.: Big bird: Transformers for longer sequences. In: Advances in Neural Information Processing Systems. vol. 33, pp. 17283–17297 (2020)
19. Zhu, L., Yu, F.R., Wang, Y., Ning, B., Tang, T.: Big data analytics in intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems **20**(1), 383–398 (2018)