Generative-Contrastive-Attentive Spatial-Temporal Network for Traffic Data Imputation

No Author Given

No Institute Given

Abstract. Data missing is inevitable in Intelligent Transportation Systems (ITSs). Although many methods have been proposed for traffic data imputation, it is still very challenging because of two reasons. First, the ground truth of missing data is actually inaccessible, which makes most imputation methods hard to train. Second, incomplete data can easily mislead the model to learn unreliable spatial-temporal dependencies, which finally hurts the imputation performance. In this paper, we proposes a novel ***G****enerative-****C****ontrastive-****A****ttentive* ***S****patial-****T****emporal* ***N****etwork* (GCASTN) model for traffic data imputation. It combines the ideas of generative and contrastive self-supervised learning together to develop a new training paradigm for imputation without relying on the ground truth of missing data. In addition, it introduces *nearest missing interval* to describe missing data and a novel ***M****issing-****A****ware* ***A****ttention* (MAA) mechanism is designed to utilize *nearest missing interval* to guide the model to adaptively learn the reliable spatial-temporal dependencies of incomplete traffic data. Extensive experiments covering three types of missing scenarios on two real-world traffic flow datasets demonstrate that GCASTN outperforms the state-of-the-art baselines.

Keywords: spatial-temporal graph data · traffic data imputation · attention · graph convolution · self-supervised learning.

1 Introduction

Traffic data collected in Intelligent Transportation Systems is usually incomplete due to machine failures, which makes them ineffective in driving the downstream deep learning models and harms the quality of corresponding services. Despite that many imputation approaches have been developed, the traffic data imputation task still has the following two challenges.

Firstly, the imputation task faces the dilemma of the absence of ground truth (label) for missing data to guide model training. Some works [20, 12] utilize the supervised learning approach to train imputation models on complete data, which is obviously not practical in real-world applications. Generative Adversarial Network (GAN) based models [13] adopt unsupervised adversarial training over the generator and the discriminator to generate complete samples to impute missing data. But they are difficult to train, as the adversarial training process

is very unstable. Variational Auto-Encoder (VAE) based models [15] learn the generation distributions of data from incomplete observed data by reconstruction, but they usually put some prior assumptions on the distributions (*e.g.*, Gaussian distribution), which is hard to be guaranteed in practice. Recently, self-supervised learning (SSL) is widely used in computer vision [8] and natural language processing [5] under the advantage that it can capture the patterns in data without requiring additional labels as the supervision information. So it is worthwhile to explore the application of SSL to solve the label-lacking problem of imputation task.

Secondly, it is difficult to learn reliable spatial-temporal dependencies in incomplete traffic data. Actually, learning the spatial-temporal dependencies is the key to modeling traffic data. And many spatial-temporal graph neural networks (STGNNs) [11, 19, 1, 7] have been proposed to effectively learn the spatial-temporal dependencies of traffic data. However, these technologies are initially designed for complete traffic data. When they are applied for imputation, they cannot be aware of the missing positions in the incomplete traffic data and cannot adjust the learned spatial-temporal dependencies adaptively. That is to say, existing spatial-temporal dependencies modeling technologies trust information from any time slice and any node, so they may capture unreliable and false spatial-temporal dependencies. Therefore, how to let the model be aware of and make full use of the missing information in incomplete traffic data so as to capture reliable spatial-temporal dependencies is challenging.

To address the above challenges, this paper proposes a novel ***G****enerative-****C****ontrastive-****A****ttentive* ***S****patial-****T****emporal* ***N****etwork* (GCASTN) for traffic data imputation. The main contributions of this paper are summarized as follows: 1) A generative-contrastive self-supervised learning is designed, which combines the ideas of generative learning and contrastive learning together, to train the traffic data imputation model without relying on the ground truth of missing data. 2) A novel missing-aware attention mechanism is proposed to capture reliable spatial-temporal dependencies from incomplete traffic data by utilizing the missing information. 3) Experiments on two real-world traffic flow datasets show that our proposed GCASTN model achieves state-of-the-art performance.

2 Related Work

Many imputation methods are proposed to deal with incomplete time series. Traditional statistical methods are the most straightforward which usually use zero values or other statistical values (*e.g.*, mean, mode, and last observation) to fill in the missing positions. Machine learning based methods, including but not limited to K-Nearest Neighbors [9], Matrix Decomposition [14], Expectation-Maximization [6] algorithm, Multivariate Imputation Chained Equations [17], Low-rank Tensor Completion [4] are also proposed for time series imputation. However, all the imputation methods mentioned above cannot effectively model the complex spatial-temporal dependencies of traffic data, therefore their imputation capabilities are limited.

The core of accurate spatial-temporal traffic data imputation lies in learning the reliable temporal and spatial dependencies of data. Recently, many deep-learning-based models show advantages in this task. BRITS[2] employs bidirectional recurrent neural networks to model temporal dependence for missing data imputation. IGNNK [18] learn the spatial dependence by graph sampling for recovering the missing data. And STGNNs that can simultaneously capture the dependencies along temporal and spatial dimensions, such as DCRNN [11], GWNet [19] and AGCRN [1], have also been applied to imputation tasks [20, 12]. However, neither of them pays attention to the bias of spatial-temporal dependencies introduced by missing data in the incomplete traffic data. In addition, GAN based models [13] whose training process is unstable and VAE based generative models [15] with unguaranteed predefined distributions are also applied to the imputation task.

3 Preliminaries

Traffic network is expressed as $\boldsymbol{\mathcal{G}} = (\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}}, \mathbf{A})$, where $\boldsymbol{\mathcal{V}}$ and $\boldsymbol{\mathcal{E}}$ are the sets of nodes (*e.g.* loop detectors or video camera) and edges respectively, $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix representing the proximity between nodes. The signals of traffic network at time slice t is defined as $\boldsymbol{X}_t = (\boldsymbol{x}_{t,1}, \boldsymbol{x}_{t,2}, \ldots, \boldsymbol{x}_{t,N})^\top \in \mathbb{R}^{N \times C}$, where C is the number of features. And we use $\boldsymbol{\mathcal{X}} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_T) \in \mathbb{R}^{N \times C \times T}$ to denote all signals over T time slices. To indicate the missing positions, we define masking matrix $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{N \times C \times T}$. For each $\boldsymbol{m}_{t,v} \in \mathbb{R}^C$ in $\boldsymbol{\mathcal{M}}$, if $\boldsymbol{x}_{t,v}$ is observable, $\boldsymbol{m}_{t,v}$ is $\mathbf{1}$, otherwise $\boldsymbol{m}_{t,v}$ is $\mathbf{0}$. And then we get corrupted signal matrix: $\tilde{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{X}} \odot \boldsymbol{\mathcal{M}}$, $\odot$ denotes element-wise dot-product. Further, we introduce the nearest missing interval $\boldsymbol{\Delta} \in \mathbb{R}^{N \times C \times T}$ that indicates the time length between the current missing position and the nearest observable one. To get any $\boldsymbol{\delta}_{t,v} \in \mathbb{R}^C$ in $\boldsymbol{\Delta}$, we first define $\boldsymbol{\delta}^h_{t,v}$ for a node v to denote the time interval between current missing position and its historical closest observable position at time slice t,

$$\boldsymbol{\delta}^h_{t,v} = \begin{cases} \mathbf{1} + \boldsymbol{\delta}^h_{t-1,v}, & t > 1 \text{ and } \boldsymbol{m}_{t,v} = \mathbf{0}, \\ \mathbf{0}, & \boldsymbol{m}_{t,v} = \mathbf{1}, \\ \mathbf{1}, & t = 1 \text{ and } \boldsymbol{m}_{t,v} = \mathbf{0}. \end{cases} \tag{1}$$

Likewise, we calculate the corresponding time interval with regard to the future nearest one $\boldsymbol{\delta}^f_{t,v}$. Finally, we get the nearest missing interval by $\boldsymbol{\delta}_{t,v} = \min(\boldsymbol{\delta}^h_{t,v}, \boldsymbol{\delta}^f_{t,v})$.

Problem Statement The task of traffic data imputation is to learn a function $f(\cdot)$ that uses the corrupted data $\tilde{\boldsymbol{\mathcal{X}}}$, masking matrix $\boldsymbol{\mathcal{M}}$ and the given graph $\boldsymbol{\mathcal{G}}$ to reconstruct complete data $\hat{\boldsymbol{\mathcal{X}}}$. Formally, $\hat{\boldsymbol{\mathcal{X}}} = f(\tilde{\boldsymbol{\mathcal{X}}}, \boldsymbol{\mathcal{M}}, \boldsymbol{\mathcal{G}})$.

4 The GCASTN model

GCASTN is an Auto-Encoder (AE) designed for traffic data imputation. It follows a novel SSL paradigm that combines generative and contrastive ideas together to train the model effectively without relying on the ground truth of

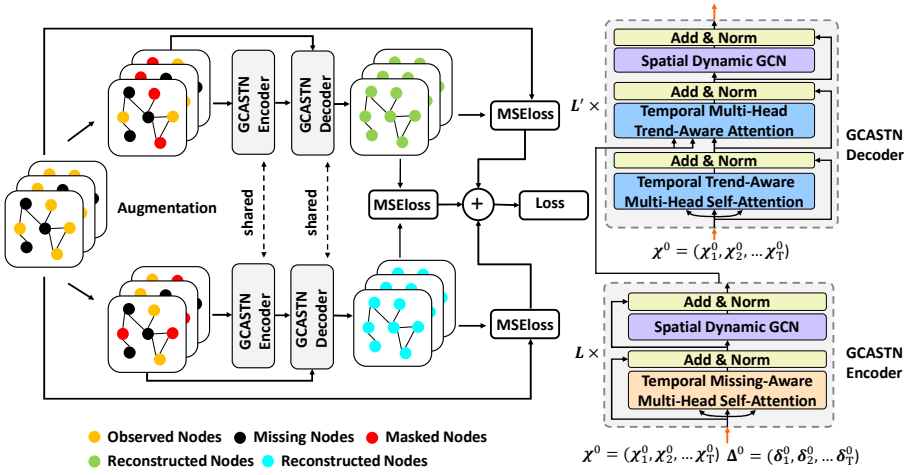

Fig. 1. The architecture of GCASTN.

missing data. Similar to classical AE, it consists of an encoder and a decoder. Specifically, the encoder uses the nearest missing interval to help the model learn the reliable spatial-temporal dependencies from incomplete data. Then the decoder reconstructs the original complete data in an autoregressive way. Figure 1 shows the overall architecture of GCASTN which is introduced in detail next.

4.1 Generative-Contrastive Self-Supervised Learning

We propose a novel learning paradigm, *i.e.*, Generative-Contrastive Learning (GCL), to train the traffic data imputation model to solve the first challenge about lacking ground truth. Actually, GCL is a kind of self-supervised learning, which combines generative and contrastive ideas together.

On the one hand, similar to generative self-supervised learning (GL), GCL is based on reconstruction tasks. Specifically, GCL first obtains its input by randomly corrupting the original incomplete data by masking. Then it tries to reconstruct the corresponding complete data via an AE. Under our traffic data imputation scenario, the original data is inherently missing, so less useful information is left after the corrupting process, which leads to the model can not utilize the spatial-temporal dependencies of traffic data well.

Thus, on the other hand, we incorporate the idea of contrastive self-supervised learning (CL) together to make up for the drawback of GL. CL learns data representations by pulling positive sample pairs together and pushing negative sample pairs away, which is helpful for learning the high-level changing trend of the traffic data. To perform contrastive learning, we need to construct positive sample pairs and negative sample pairs. Specifically, with regard to the positive sample pairs, we randomly mask the original incomplete input twice, try to respectively reconstruct the complete data through the same AE, and finally align these

two recovered data. Meanwhile, constructing negative pairs is necessary to avoid the mode collapse. However, it is not easy and needs careful design. To relieves the stress of manually designing negative pairs constructing strategies, we treat the above generative self-supervised learning part as an alternative way since the reconstruction learning naturally pushes different samples away. The overall pipeline works as follows.

Firstly, our model takes the original corrupted signal matrix $\tilde{\boldsymbol{\mathcal{X}}}$ and its corresponding masking matrix $\boldsymbol{\mathcal{M}}$ as inputs. Then, we make data augmentation through a two-fold cross random masking strategy (see details in Section 4.2), and the two augmented inputs are denoted as $\tilde{\boldsymbol{\mathcal{X}}}^{'}$ and $\tilde{\boldsymbol{\mathcal{X}}}^{''}$. After that, we input both the two augmented inputs into the same AE that consists of an encoder and a decoder to reconstruct the complete traffic data, denoted as $\hat{\boldsymbol{\mathcal{X}}}^{'}$ and $\hat{\boldsymbol{\mathcal{X}}}^{''}$ respectively. Finally, defining α as a hyperparameter, the training loss $\mathcal{L} = \mathcal{L}_{\mathrm{G}} + \alpha\mathcal{L}_{\mathrm{C}}$ consists of two parts. One is the reconstruction loss for GL, *i.e.*, $\mathcal{L}_{\mathrm{G}} = \frac{1}{\|\boldsymbol{\mathcal{M}}\|_0} \sum (\|\hat{\boldsymbol{\mathcal{X}}}^{'} - \tilde{\boldsymbol{\mathcal{X}}}\|_2^2 + \|\hat{\boldsymbol{\mathcal{X}}}^{''} - \tilde{\boldsymbol{\mathcal{X}}}\|_2^2) \odot \boldsymbol{\mathcal{M}}$. The other one is the contrastive loss for CL, *i.e.*, $\mathcal{L}_{\mathrm{C}} = \frac{1}{\|\boldsymbol{\mathcal{M}}\|_0 + \|\mathbf{1} - \boldsymbol{\mathcal{M}}\|_0} \sum (\|\hat{\boldsymbol{\mathcal{X}}}^{'} - \hat{\boldsymbol{\mathcal{X}}}^{''}\|_2^2)$, which aligns positive pairs.

4.2 Data Augmentation via Two-Fold Cross Random Masking

Two-fold cross random masking is a data augmentation strategy specially designed for GCL, which generates a pair of positive samples for an input. The significant advantage of this data augmentation strategy is that all the observable data in the original input are remained to be utilized.

In detail, we first get two all-zero matrices $\tilde{\boldsymbol{\mathcal{X}}}^{'}$ and $\tilde{\boldsymbol{\mathcal{X}}}^{''}$ of the same size as $\tilde{\boldsymbol{\mathcal{X}}}$. Then observed values in $\tilde{\boldsymbol{\mathcal{X}}}$ will replace the zero values at the corresponding identical positions in $\tilde{\boldsymbol{\mathcal{X}}}^{'}$ with the probability at 50%. If an observable value does not exist in $\tilde{\boldsymbol{\mathcal{X}}}^{'}$, it must appear in the same position in $\tilde{\boldsymbol{\mathcal{X}}}^{''}$ to guarantee no useful information loss. Formally,

$$\begin{aligned} &\{\boldsymbol{x}_{t,v} | \boldsymbol{x}_{t,v} \in \tilde{\boldsymbol{\mathcal{X}}}^{'}\} \cup \{\boldsymbol{x}_{t,v} | \boldsymbol{x}_{t,v} \in \tilde{\boldsymbol{\mathcal{X}}}^{''}\} = \{\boldsymbol{x}_{t,v} | \boldsymbol{x}_{t,v} \in \tilde{\boldsymbol{\mathcal{X}}}\} \\ &\{\boldsymbol{x}_{t,v} | \boldsymbol{x}_{t,v} \in \tilde{\boldsymbol{\mathcal{X}}}^{'}\} \cap \{\boldsymbol{x}_{t,v} | \boldsymbol{x}_{t,v} \in \tilde{\boldsymbol{\mathcal{X}}}^{''}\} = \{\mathbf{0}\} \end{aligned} \tag{2}$$

Through the above data augmentation, we get two augmented signal matrices $\tilde{\boldsymbol{\mathcal{X}}}^{'}$ and $\tilde{\boldsymbol{\mathcal{X}}}^{''}$ as the positive sample pair.

4.3 GCASTN Encoder

The encoder in our model aims to encode the incomplete input into hidden space by discovering and exploiting the reliable spatial-temporal dependencies in traffic data. Specifically, it contains a stack of L identical spatial-temporal blocks (ST-blocks), and each ST-block includes a temporal Missing-Aware Multi-head Self-Attention (MissMultiSA) block and a spatial Dynamic Graph Convolution Network (DGCN) [7] block with residual connection and layer normalization

[16]. Compared to existing technologies, our ST-blocks are able to learn reliable spatial-temporal dependencies from missing data.

Specifically, the input to our encoder includes an augmented sample and its corresponding nearest missing interval. Firstly, we linearly project the augmented sample into a high-dimensional tensor with size d_{model} and then add temporal position embedding [16] and spatial position embedding [7] to obtain $\boldsymbol{\mathcal{X}}^0$. Meanwhile, we obtain $\boldsymbol{\Delta}^0$ by linear mapping the nearest missing interval. Finally, the input to the l^{th} block in our encoder is the output of the $(l-1)^{th}$ block $\boldsymbol{\mathcal{X}}^{l-1} \in \mathbb{R}^{N \times d_{\text{model}} \times T}$ along with $\boldsymbol{\Delta}^0$.

Temporal Missing-Aware Multi-Head Self-Attention MissMultiSA is a self-attention mechanism specially designed to modelling incomplete data. It makes use of the nearest missing interval information to perceive the missing positions in traffic data, and then adaptively adjusts the learned spatial-temporal dependencies to make the learning process of the model more reliable.

MissMultiSA is the multi-head version of Missing-Aware Attention (MAA). Unlike the widely used Scaled Dot-Product Attention (SDPA) [16], MAA employs two query-key pairs to measure the correlations between the queries and keys. Specifically, MAA in the l^{th} block uses $\boldsymbol{\mathcal{X}}^{l-1}$, *i.e.*, the hidden representations for incomplete traffic data as both the queries $\boldsymbol{Q}$, keys $\boldsymbol{K}$ and values $\boldsymbol{V}$ to perform self-attentions. Considering the computational efficiency, MAA first utilizes Scaled Dot-Product (SDP) to get the weights between queries and keys. SDP is defined as:

$$\text{SDP}(\boldsymbol{Q}, \boldsymbol{K}) = \text{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top / \sqrt{d_{\text{model}}}) \tag{3}$$

However, directly using the above weights to measure the correlations between the elements in data, the model can neither perceive the existence of the missing values nor measure the effect brought by the missing values. This leads to model to capture unreliable dependencies and even to aggregate false information from missing positions. Hence, MAA further utilizes another query-key pairs from the nearest missing interval $\boldsymbol{\Delta}^0$ to get additional weights to revise the former weights. The new weights help the model to accurately perceive the missing information in data, and let the model to consider the corresponding influences. Finally, the output is computed as a weighted sum of the values $\boldsymbol{V}$.

Foramlly, let $\boldsymbol{Q}_\Delta$ and $\boldsymbol{K}_\Delta$ denote additional queries and keys from the nearest missing interval $\boldsymbol{\Delta}^0$, MAA is expressed as:

$$\text{MAA}(\boldsymbol{Q}_\Delta, \boldsymbol{K}_\Delta, \boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = (\text{SDP}(\boldsymbol{Q}_\Delta, \boldsymbol{K}_\Delta) \odot \text{SDP}(\boldsymbol{Q}, \boldsymbol{K}))\boldsymbol{V} \tag{4}$$

Then we follows ASTGNN [7] to introduce 1D convolutions in the temporal dimension to learn the local trend information, and extend MAA to multi-head for learning richer information [16], Finally, we get MissMultiSA as follows:

$$\begin{aligned} &\text{MissMultiSA}(\boldsymbol{Q}_{\boldsymbol{\Delta}}, \boldsymbol{K}_{\boldsymbol{\Delta}}, \boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \oplus(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)\boldsymbol{W}^O \\ &\text{head}_j = \text{MAA}(\boldsymbol{Q}_{\boldsymbol{\Delta}}\boldsymbol{W}_j^Q, \boldsymbol{K}_{\boldsymbol{\Delta}}\boldsymbol{W}_j^K, \boldsymbol{\Phi}_j^Q \star \boldsymbol{Q}, \boldsymbol{\Phi}_j^K \star \boldsymbol{K}, \boldsymbol{V}\boldsymbol{W}_j^V) \end{aligned} \tag{5}$$

where $\boldsymbol{W}^O$, $\boldsymbol{W}_j^Q$, $\boldsymbol{W}_j^K$ and $\boldsymbol{W}_j^V$ are the learnable parameters, $\boldsymbol{\Phi}_j^Q$ and $\boldsymbol{\Phi}_j^Q$ are the parameters of convolution kernels, while $\star$ is the convolution operation.

Spatial Dynamic Graph Convolution Network DGCN [7] is a graph neural network for learning spatial dependencies of traffic data. With the aid of the attention mechanism, DGCN can dynamically adjust the correlation strengths among nodes and better aggregate node information, which learn about more reasonable spatial dependencies. Denote the output of MissMultiSA in the l^{th} block of encoder as $\boldsymbol{\mathcal{Z}}^{l-1} = (\boldsymbol{Z}_1^{l-1}, ..., \boldsymbol{Z}_T^{l-1}) \in \mathbb{R}^{N \times d_{model} \times T}$. DGCN calculates the spatial relevant weights $\boldsymbol{S}_t^l$ of the l^{th} block via $\boldsymbol{S}_t^l = \mathrm{SDP}(\boldsymbol{Z}_t^{l-1}, \boldsymbol{Z}_t^{l-1}) \in \mathbb{R}^{N \times N}$. And then it uses the spatial relevant weights to adjust the static normalized adjacency matrix $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$, where $\boldsymbol{D}$ is the degree matirx of the traffic network. Finally DGCN aggregates node information to obtain a new node representation. The whole computation process of DGCN can be defined formally:

$$\boldsymbol{X}_t^l = \sigma((\tilde{\boldsymbol{A}} \odot \boldsymbol{S}_t^l)\boldsymbol{Z}_t^{l-1}\boldsymbol{W}^l) \tag{6}$$

where σ is activation function and $\boldsymbol{W}^l$ is learnable parameters.

4.4 GCASTN Decoder

The decoder aims to decode the learned spatial-temporal dependencies for reconstructing the complete signal matrices. Like the encoder, the decoder stacks $L^{'}$ blocks, and each block contains two temporal attention modules and a DGCN module with residual connection and layer normalization. The first temporal attention module encodes the hidden representation of missing data into an output, and the output is used as the queries of the second temporal attention module. While the corresponding keys and values are obtained from the encoder's output $\boldsymbol{\mathcal{X}}^L$. Different from the operations in the encoder, in our decoder the two temporal attention modules of each block replace MAA with SDPA. After that, the decoder further employs DGCN again to decode the spatial dependencies. Figure 1 shows the details of the encoder and decoder in our model.

5 Experiments

5.1 Datasets and Baselines

We use two real traffic flow datasets including PEMS04 and PEMS08 to evaluate our proposed method. There are 307 nodes in PEMS04 and 170 nodes in PEMS08. They are collected through California Transportation Agencies Performance Measurement System (PeMS) [3] from 1 January 2018 and 1 July 2016 respectively, which span two months. Initially they are collected every 30 seconds, and then we aggregate the raw data into 5-minute interval. Next we construct corrupted datasets with different missing types and missing rates on the two complete datasets. Here we consider three types of absence: random

missing (RM) [4], non-random missing (NM) [4] and block missing (BM) [12]. RM means that data may be missing randomly at any time slice and node. NM is relational in temporal dimension and independent in spatial dimension. It occurs when any single collector does not work within a period (e.g. one day). BM is correlated in all dimensions, indicating failure of collectors in an area for a while. In particular, once the data is missing, all features collected by the collector are missing. For missing rates, we set 20%, 30%, 70% and 90% corresponding to low and high missing rates. And all corrupted datasets are generated by referring to prior works [4, 12].

Table 1. Imputation performances on PEMS04.

model (RM/NM/BM))	20%			30%			70%			90%		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
last (RM)	21.42	34.67	14.63*	21.81	34.53	14.93*	25.34	41.09*	17.54	38.56	62.67	29.68
DCRNN (RM)	21.70	32.38	19.13	24.11	34.69	25.61	47.32	70.60	53.86	78.93	110.03	126.34
GWNet (RM)	38.06	55.02	23.51	38.28	57.00	25.00	63.33	92.79	34.18	128.67	183.02	63.03
AGCRN (RM)	22.42	34.23	15.64	25.23	38.73	17.08	35.81	63.80	21.62	75.58	135.20	37.16*
LATC (RM)	19.63*	31.70*	15.16	20.08*	32.48*	15.61	28.76	45.38	21.37	61.60	87.36	67.58
IGNNK (RM)	20.60	32.86	19.18	21.16	33.55	17.85	24.10	38.15	22.81	42.02*	65.22*	43.98
BRITS (RM)	23.13	40.63	20.72	25.14	43.87	23.81	39.46	61.35	45.89	71.39	100.01	97.93
MTAN (RM)	18.36	29.66	13.32	19.27	31.31	13.25	24.21*	41.99	18.18*	58.53	100.48	66.81
GCASTN (RM)	**15.44**	**25.04**	**10.80**	**15.47**	**25.28**	**11.18**	**17.55**	**28.25**	**12.43**	**20.67**	**33.02**	**16.12**
last (NM)	21.73	35.23	14.62	21.81	35.44	14.90*	25.57*	41.02*	17.73	38.10	61.11*	29.07
DCRNN (NM)	21.72	32.62	18.07	23.16	33.85	24.04	30.18	48.68	29.71	70.33	101.61	111.97
GWNet (NM)	29.44	45.68	22.40	35.01	51.67	22.02	68.63	99.14	36.46	119.61	155.02	58.48
AGCRN (NM)	19.12	30.54	14.38*	20.00*	31.08	15.12	34.30	63.11	22.11	79.65	137.17	39.89*
LATC (NM)	40.34	66.31	21.87	62.58	95.91	32.00	138.64	187.81	67.67	176.56	228.98	82.82
IGNNK (NM)	21.17	33.76	18.60	21.60	33.70	19.63	24.25	38.03	21.95	39.85*	59.64	43.04
BRITS (NM)	27.30	45.27	31.06	28.19	46.14	35.84	43.62	64.45	63.10	81.19	105.61	121.96
MTAN (NM)	19.19*	30.95*	13.92	19.17	31.14*	13.38	29.60	67.85	21.41*	94.91	165.32	61.46
GCASTN (NM)	**15.75**	**25.79**	**10.90**	**15.58**	**25.41**	**11.29**	**17.76**	**28.59**	**14.08**	**20.91**	**33.36**	**16.60**
last (BM)	26.65	42.75	18.69*	28.66	46.08	20.34*	57.46	92.23	52.57	124.75	170.64	170.36
DCRNN (BM)	20.84	31.34	17.91	22.02	32.96	19.05	28.75	44.12	30.32*	69.76*	102.35*	115.36
GWNet (BM)	22.60*	32.02*	21.65	25.10*	36.52*	20.36	46.94	65.29	28.83	98.03	120.40	**47.72**
AGCRN (BM)	39.37	71.87	26.64	47.69	85.81	32.07	109.15	174.84	51.30	107.35	132.23	159.26
LATC (BM)	27.07	42.74	20.11	32.84	50.11	23.51	43.46	69.33	35.23	93.16	138.74	55.16
IGNNK (BM)	34.41	58.63	42.44	40.52	67.77	57.51	82.56	118.36	128.98	119.38	149.87	209.62
BRITS (BM)	27.27	44.70	33.72	28.56	41.78	29.64	29.33*	47.74*	34.73	**61.68**	**83.60**	89.79
MTAN (BM)	38.67	82.23	24.63	43.82	88.24	32.53	90.01	138.92	99.01	120.94	152.39	210.04
GCASTN (BM)	**17.91**	**29.22**	**12.55**	**18.75**	**30.34**	**13.43**	**24.65**	**40.57**	**17.75**	63.77	94.56	81.50*

Also we need some baselines. Briefly introduce them: 1) **BRITS** [2] employs bidirectional recurrent neural networks to impute missing data. 2) **mTAN** [15] A VAE based model designs an attention mechanism that can map the missing sequences into a fixed-length representation for imputation. 3) **IGNNK** [18] learn the spatial dependence by graph sampling for recovering the incomplete data. 4) **LATC** [4] is the latest low-rank matrix/tensor completion method. 5) **LAST** simply uses the last observed value to replace the missing value. 6)-8) **DCRNN** [11],**GWNet** [19] and **AGCRN** [1] are Spatial-Temporal Graph Neural Networks which can learn spatial-temporal dependencies for imputation.

5.2 Experimental Results

Settings We divide all datasets into training and validation sets at ratio 8:2 by the time and use Min-Max method to normalize all data, then we construct samples by sliding window on the time axis. Where we set the window length T to 12 and the sliding step size to 1. During training, the information of observable and masked positions in the training sets are used as supervision. To prevent overfitting, the observable values in the validation sets are masked off with 10% probability at random. In the evaluation stage, we only evaluate the imputation results on missing positions. And the ground truth of missing positions is only known during the evaluation stage. The evaluation metrics in this paper are mean absolute error (MAE), root mean square error (RMSE) and the mean absolute percentage error (MAPE).

Table 2. Imputation performances on PEMS08.

model (RM/NM/BM)	20%			30%			70%			90%		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
last (RM)	16.86	26.51	10.46*	17.14*	27.00*	10.63	20.39	32.26	12.59	31.70*	51.60*	20.21*
DCRNN (RM)	29.42	40.59	22.92	36.67	48.68	30.53	39.04	58.69	41.28	73.82	101.86	123.81
GWNet (RM)	30.88	45.05	17.01	30.29	44.36	16.45	54.14	81.82	25.54	111.58	150.71	48.64
AGCRN (RM)	17.65	27.13	12.46	18.26	28.08	12.22	28.26	50.14	17.24	63.53	112.38	30.07
LATC (RM)	14.42*	23.53*	10.66	14.48	23.70	10.67*	17.04	27.54	12.56*	21.67	34.51	16.35
IGNNK (RM)	17.72	26.23	20.86	20.05	29.28	33.44	21.50	32.32	28.57	41.99	63.75	65.45
BRITS (RM)	22.91	37.03	30.88	32.57	46.77	51.94	45.25	62.55	72.18	74.04	96.24	133.00
MTAN (RM)	13.99	22.02	9.39	18.64	28.99	11.58	18.33*	30.24*	11.61	57.02	102.32	56.47
GCASTN (RM)	**11.91**	**19.26**	**7.90**	**12.13**	**19.62**	**8.12**	**13.45**	**21.99**	**9.18**	**17.64**	**28.48**	**12.27**
last (NM)	16.89	26.34	10.64*	17.08	26.91	10.90	20.41	32.27	12.57	31.12	50.02	19.8
DCRNN (NM)	16.67	24.85	13.19	17.90	26.35	17.94	39.10	58.45	47.26	63.26	92.40	93.71
GWNet (NM)	29.01	40.15	19.09	32.83	46.93	18.21	54.45	80.74	25.31	109.75	143.57	45.89
AGCRN (NM)	14.47	22.22	10.67	14.61	22.69	10.66*	26.55	48.29	15.38	71.87	126.01	33.00*
LATC (NM)	15.62	25.45	11.70	15.84	25.99	12.04	26.09	54.63	16.99	112.54	171.73	51.97
IGNNK (NM)	17.88	26.16	25.71	19.28	27.91	26.60	22.20*	32.56*	27.03	38.48*	56.08*	55.96
BRITS (NM)	24.86	39.69	38.00	25.50	40.57	40.80	45.58	65.27	77.59	91.78	112.37	167.44
MTAN (NM)	14.51*	22.45*	9.82	14.91*	23.35*	10.01	22.22	48.57	13.22*	119.13	206.23	80.74
GCASTN (NM)	**11.54**	**18.75**	**7.73**	**11.94**	**19.39**	**8.30**	**13.75**	**22.34**	**9.61**	**17.53**	**28.44**	**11.94**
last (BM)	21.67*	34.71*	13.38*	23.36*	37.40*	14.40*	48.73*	78.87*	34.43*	105.60	147.69	87.95
DCRNN (BM)	75.23	97.26	113.50	58.43	84.37	79.85	88.31	118.51	165.65	114.14	140.72	233.71
GWNet (BM)	59.66	85.37	31.52	54.67	78.36	32.04	126.29	160.01	55.53	157.11	184.54	63.41
AGCRN (BM)	33.42	61.29	20.93	42.18	78.42	22.78	86.62	138.43	37.57	90.27*	112.81	80.75
LATC (BM)	17.04	27.20	12.43	17.68	28.14	12.57	34.78	55.62	22.07	144.90	193.19	64.66*
IGNNK (BM)	29.95	50.20	42.44	36.53	60.47	48.98	75.86	105.87	142.32	112.01	128.26	228.08
BRITS (BM)	52.91	70.99	96.90	58.12	76.79	97.45	70.89	92.36	113.93	89.92	112.94*	169.19
MTAN (BM)	40.23	91.88	20.97	49.93	106.28	26.19	89.64	137.05	97.30	115.15	148.16	182.71
GCASTN (BM)	**14.49**	**23.75**	**9.65**	**15.28**	**25.46**	**10.42**	**22.63**	**38.46**	**15.69**	**60.25**	**88.34**	**49.43**

We implemented our GCASTN model in the PyTorch[1] framework. The number of layers for both decoder and encoder is 4, d_{model} is 64. About MissMultiSA, the number of attention heads is 8 and the convolution kernel size is 3. About Training, we use Adam [10] algorithm to train networks, learning rate is 0.001. All experiments are conducted on a NVIDIA RTX A4000 GPU.

[1] https://pytorch.org

Overall Performance The results of the experiment are shown in Tables 1 and 2. And in each corrupted dataset, The best results are marked in bold, the second place results are underlined, and the third place results are marked by '*'.

As we can see, our proposed method (GCASTN) achieves the best results for most of missing cases. In the case of RM, NM, GCASTN performs better than the baseline methods by a large margin. However, when the missing rate is 90% of the BM case, the imputation of all models is not very satisfactory, and even GCASTN cannot outperform other models in this case for PEMS04 data. In addition, the results in each row show that all models show a decreasing trend in imputation performance as the missing rate increases (Less observable values), and baselines degrades at a much faster rate, while GCASTN drops slower. Finally, in terms of imputation performance under different missing types, the BM case is the most difficult to interpolate, and NM is slightly harder than RM.

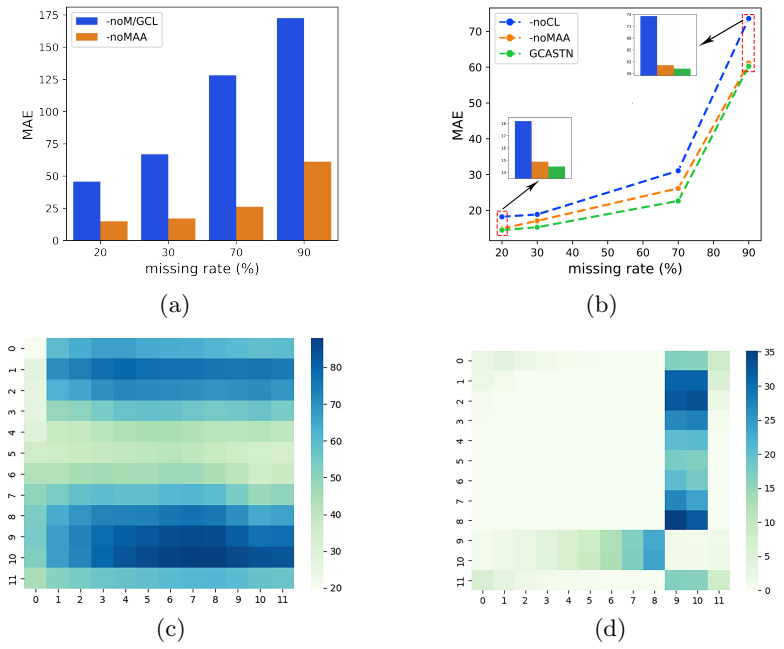

Fig. 2. The results of ablation experiments on PEMS08.

Overall, even with limited observable data, GCASTN receives less influence from missing positions and can learn reasonable spatial-temporal dependencies for imputation.

Ablation experiments In this section we evaluate the two innovations we propose. First we design three variants of **GCASTN** for comparison:

- **-noCL**: It trains the model without the contrastive loss.
- **-noMAA**: It replaces the MAA with SDPA [16].
- **-noM/GCL**: It uses SDPA [16] instead of MAA and does not trains models with generative contrast learning.

All settings of the variant models are consistent with GCASTN except the differences mentioned above. Due to space limitations, we only conduct ablation experiments on the PEMS08 dataset and only calculate MAE. The results are shown on Figure 2.

First we evaluate the effects of generative-contrastive learning. By comparing -noMAA and -noM/GCL from Figure 2(a), we can see our proposed GCL has tremendous advantages when applied to imputation tasks. With the help of the GCL, general spatial-temporal graph neural networks can also achieve superior results in imputation tasks. For Further exploring the effectiveness of the GCL, we get the experimental results of -noCL and GCASTN and show them on Figure 2(b). Obviously, GCL incorporating contrastive learning ideas can effectively improve the imputation accuracy.

Secondly we assess the Missing-Aware Attention. On Figure 2(b), GCASTN performs better than -noMAA, which indicates MMA can learn more reliable spatial-temporal dependencies by using nearest missing intervals. To demonstrate the strengths of MAA more visually, the attention scores between locations calculated via -noMAA and GCASTN for the same corrupted input sequence are shown in Figures 2(c) and 2(d) respectively. Comparing Figures 2(c) and 2(d), it can be seen that GCASTN can clearly perceive missing positions and correctly aggregates reliable information.

6 Conclusion

In this paper, we propose a novel self-supervised method GCASTN for missing traffic data imputation. We combine generative and contrastive methods in self-supervised learning to create the generative-contrastive learning, which solve the dilemma of the absence of ground truth (label) for missing data. In addition, we introduce the nearest missing interval and design missing-aware attention to perceive missing positions for capturing reliable spatial-temporal dependencies. And extensive experimental results demonstrate the superiority of GCASTN.

References

1. Bai, L., Yao, L., Li, C., Wang, X., Wang, C.: Adaptive graph convolutional recurrent network for traffic forecasting. arXiv preprint arXiv:2007.02842 (2020)
2. Cao, W., Wang, D., Li, J., Zhou, H., Li, L., Li, Y.: Brits: Bidirectional recurrent imputation for time series. arXiv preprint arXiv:1805.10572 (2018)

3. Chen, C., Petty, K., Skabardonis, A., Varaiya, P., Jia, Z.: Freeway performance measurement system: mining loop detector data. Transportation Research Record **1748**(1), 96–102 (2001)
4. Chen, X., Lei, M., Saunier, N., Sun, L.: Low-rank autoregressive tensor completion for spatiotemporal traffic data imputation. arXiv preprint arXiv:2104.14936 (2021)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
6. García-Laencina, P.J., Sancho-Gómez, J.L., Figueiras-Vidal, A.R.: Pattern classification with missing data: a review. Neural Computing and Applications **19**(2), 263–282 (2010)
7. Guo, S., Lin, Y., Wan, H., Li, X., Cong, G.: Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. IEEE Transactions on Knowledge and Data Engineering (2021)
8. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. arXiv preprint arXiv:2111.06377 (2021)
9. Hudak, A.T., Crookston, N.L., Evans, J.S., Hall, D.E., Falkowski, M.J.: Nearest neighbor imputation of species-level, plot-scale forest structure attributes from lidar data. Remote Sensing of Environment **112**(5), 2232–2245 (2008)
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
11. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. arXiv preprint arXiv:1707.01926 (2017)
12. Liang, Y., Zhao, Z., Sun, L.: Dynamic spatiotemporal graph convolutional neural networks for traffic data imputation with complex missing patterns. arXiv preprint arXiv:2109.08357 (2021)
13. Luo, Y., Zhang, Y., Cai, X., Yuan, X.: E2gan: End-to-end generative adversarial network for multivariate time series imputation. In: AAAI Press. pp. 3094–3100 (2019)
14. Morup, M., Dunlavy, D.M., Acar, E., Kolda, T.G.: Scalable tensor factorizations with missing data. Tech. rep., Sandia National Laboratories (2010)
15. Shukla, S.N., Marlin, B.M.: Multi-time attention networks for irregularly sampled time series. arXiv preprint arXiv:2101.10318 (2021)
16. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
17. White, I.R., Royston, P., Wood, A.M.: Multiple imputation using chained equations: issues and guidance for practice. Statistics in medicine **30**(4), 377–399 (2011)
18. Wu, Y., Zhuang, D., Labbe, A., Sun, L.: Inductive graph neural networks for spatiotemporal kriging. arXiv preprint arXiv:2006.07527 (2020)
19. Wu, Z., Pan, S., Long, G., Jiang, J., Zhang, C.: Graph wavenet for deep spatial-temporal graph modeling. arXiv preprint arXiv:1906.00121 (2019)
20. Yang, B., Kang, Y., Yuan, Y., Huang, X., Li, H.: St-lbagan: Spatio-temporal learnable bidirectional attention generative adversarial networks for missing traffic data imputation. Knowledge-Based Systems **215**, 106705 (2021)