Complex & Intelligent Systems (2023) 9:1363–1376
https://doi.org/10.1007/s40747-022-00856-w

ORIGINAL ARTICLE

A multi-task learning-based generative adversarial network for red tide multivariate time series imputation

Longfei Xu[1] · Lingyu Xu[1] · Jie Yu[1]

Received: 15 November 2021 / Accepted: 16 August 2022 / Published online: 7 September 2022

Abstract
Red tide data are typical multivariate time series (MTS) and complete data help analyze red tide more conveniently. However, missing values due to artificial or accidental events hinder further analysis of red tide phenomenon. Generative adversarial network (GAN) is effective in capturing distribution of MTS while the imputation performance is far from satisfactory, especially in conditions of high missing rate. One of the remaining open challenges is that common GAN-based imputation methods usually lack the ability to excavate implicit correlations between different attributions and downstream tasks, from which advanced latent information about missing values can be mined to improve imputation performance. To deal with the problem, a novel multi-task learning-based generative adversarial imputation network (MTGAIN) is proposed by introducing the prediction task into GAN to unearth more detailed information about missing values to better model distribution of red tide MTS. Furthermore, the homoscedastic uncertainty of multiple tasks is exploited to balance the weights of losses between generation and prediction tasks. The experiments conducted on a real-world dataset demonstrate that MTGAIN outperforms existing methods in terms of imputation and post-imputation performances, especially in conditions of high missing rate.

Keywords Red tide · Imputation · Generative adversarial network · Multi-task learning

Introduction

MTS is one of the most common and important data formats. Complete MTS contains rich temporal dependencies between different time intervals as well as intimate relationships among different attributions [1]. Various applications of MTS include meteorological prediction, fault diagnosis, financial analysis, traffic flow adjustment, etc. [2].

Red tide data are a typical form of MTS. Red tide analysis requires complete and detailed datasets which will contribute to the exhaustive understanding of the red tide. However, the phenomenon of missing values in red tide MTS is almost inevitable and poses a crucial challenge for related researches. Plenty of reasons will give rise to the problem, such as malfunctions in data collection, anomalies in transmission procedure, device failures in machine operation, etc. [3]. Missing values in MTS will not merely result in a serious deviation between complete and incomplete data but also bring about poor model performance and more complicated analysis in real-world applications [4]. Consequently, missing values in red tide MTS is deemed an urgent problem to be carefully addressed with practical methods for the subsequent downstream analysis.

The deletion and imputation are among the most popular methods to handle missing values [5]. Deletion is to delete the entire attributions if any of the samples has a missing value in the corresponding attribution and reserve the ones with complete values. In spite of the convenience and simple operation, the deleted attributions may contain significant latent patterns and dependencies, making the deletion not a suitable strategy, especially in conditions of high missing rate.

The imputation method is a more widely accepted way to deal with missing values with the purpose of using existing information under observations to recover original data in the pre-processing step [6]. Many analysis methodologies of MTS can be applied to the recovered data after imputation [7]. Traditional statistical methods attempt to impute missing values through statistical properties of MTS. Mean and median methods conduct imputation, respectively, by replac-

✉ Longfei Xu
18851310017@163.com

[1] School of Computer Engineering and Science, Shanghai University, Shanghai, China

ing the missing values with mean and median values of the observed data [8]. However, statistical imputation methods usually omit the temporal relationships among different time intervals and ignore the variance of missing values, which may introduce outliers and change statistical characteristics of the original data.

Machine learning imputation methods utilize some properties of MTS through complicated ways to finish imputation process. Popular methods include K-nearest neighbor (KNN) [9], singular value decomposition (SVD) [10] and expectation maximization (EM) [11]. KNN attempts to calculate the similarity distance between the given samples and other samples through distance measurements like Euclidean distance. This aims to identify corresponding samples which are similar to the incomplete samples. However, it is inevitable for KNN to search the whole dataset to find samples that meet the requirements, which is a time-consuming procedure, especially in conditions of large datasets. SVD methods carry out singular value decomposition of the complete data matrix and specific the number of singular values to be retained. This process will restore an approximate matrix through the eigenmatrix corresponding to the retained singular values. The values of approximate matrix are imputed in the positions where corresponding missing value takes place in the original matrix. It is a time-consuming process for SVD to repeatedly conduct new decomposition for each missing sample, resulting in SVD inefficient on large matrices. EM algorithm is an iterative method to calculate maximum likelihood estimation or posterior distribution to impute missing values. However, EM methods may fall into the local extremum and converge slowly. The above methods fail to fully explore the temporal dependencies to impute accurate values, which means they tend to lose efficiency because the temporal dependencies contained in MTS accounts for a large proportion.

Recently, plenty of deep learning methods have been applied to various fields including MTS analysis, image processing and speech recognition, etc. Deep learning-based imputation methods have been proposed and shown great potential [12]. Recurrent neural network (RNN) [13] and auto-encoder (AE) [14] are among the most popular as well as relevant variants of them. RNN methods utilize recursive connection to memorize the temporal dependencies between different time intervals which are vital for the reconstruction of MTS. AE learns a compressed representation of complete data by bottleneck layers to reserve the important characteristic to reconstruct original data. However, AE methods may lose generalization without any constraints, which makes it hard to operate well given new samples. Variational auto-encoder (VAE) [15] is probabilistic AE designed to find a low-dimensional representation of real data. VAE has the ability to produce realistic fake data by constraining the form of latent space distribution [16]. Uses the VAE to approximate the probability distribution of the traffic data based on the assumption that traffic data can be generated from a low-dimensional latent space. The heterogeneous-incomplete VAE (HI-VAE) [17] extends the vanilla VAE to handle incomplete and heterogeneous data. It aims to learn the correlations between different attributes through a Gaussian mixture to span a latent space.

GAN [18] is a more appropriate option to model data distribution compared with VAE. As a class of generative models, GAN specializes in learning a mapping from latent space to the real data distribution. Deep convolutional generative adversarial network (DCGAN) [19] is composed of various deep convolutional neural networks and good at handling 2-D data with spatial regularities. DCGAN suffers from dealing with MTS which have no such spatial regularities. To handle the limitations of DCGAN, multivariate time series generative adversarial network (MTS-GAN) [20] reconstructs missing values by replacing 2-D convolution in DCGAN with multi-channel 1-D convolution to better capture the characteristics of MTS. GAN-2-stage [21] conducts data imputation through the time lag matrix considering that the time dependencies between missing values and recently observed values should decay with the increase of time interval. It tries to find the optimal noise vectors to generate synthetic data with a 2-part loss including masked reconstruction loss and discriminative loss and thus leads to poor time efficiency. Compared with [21], end-to-end GAN (E^2GAN) [22] takes a compressing and reconstructing strategy to avoid the noise optimization stage. E^2GAN can generate reasonable missing values at one stage and gain better time efficiency than multi-stage methods. Generative adversarial imputation nets (GAIN) [23] exploits the standard GAN architecture and operates well when complete data are unavailable. The generator conducts the imputation process and the discriminator is trained to distinguish imputed values from original values. GAIN introduces the hint mechanism to provide partial information for the discriminator to confirm the generator has learned the real data distribution. The hint mechanism is also exploited in [24] with the modified RNN to capture temporal dependencies across time steps. To alleviate the interference of local clutter and the inaccurate imputation boundary details, Generative Adversarial Guider Imputation Network (GAGIN) [25] designed different components to incorporate local and global results from rough to accurate.

More recently, some works exploit graph convolutional network (GCN) to deal with the imputation task. Gated attentional GAN (GaGAN) [26] combines the GCN and the gate recurrent unit to, respectively, capture spatial and temporal correlation for signalized road networks. The self-attention mechanism is applied to better model traffic patterns. Graph imputer neural network (GINN) [27] frames the imputation problem in terms of a GCN auto-encoder. The GCN encoder encodes data into the intermediate embedding which

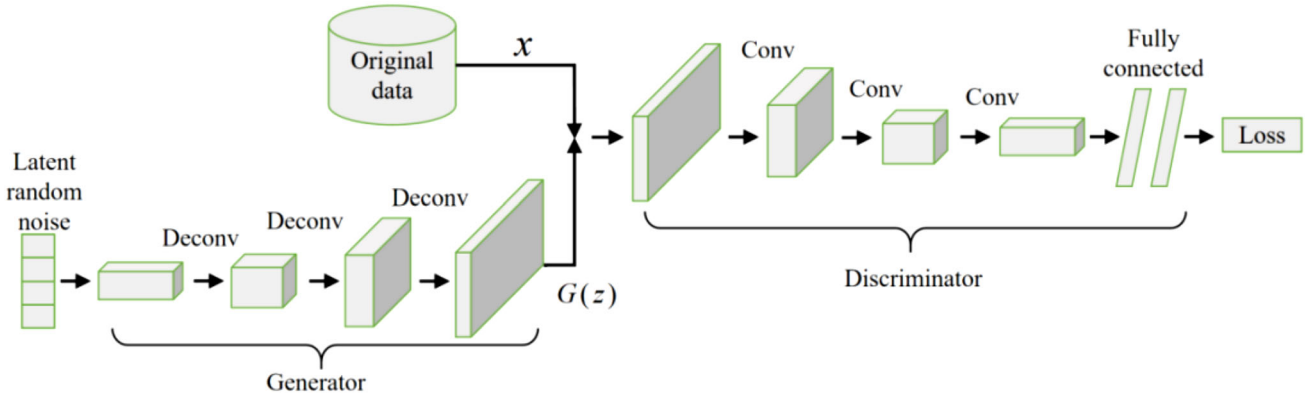

Fig. 1 The architecture of vanilla GAN

is used to reconstruct imputed data by another GCN decoder. GCNMF [28] uses Gaussian mixture distributions to represent incomplete features and derive the expected activation of the first layer neurons in GCN. Other graph-based methods focus on traffic data imputation including [29–31].

These imputation methods merely try to find information about missing values in local generation tasks. They tend to ignore the fact that implicit correlations between multiple attributions and downstream tasks usually contain more detailed information about the missing values, which can be unearthed by multi-task learning methods [32]. This inspires us to introduce the idea of multi-task learning into GAIN to handle the limitations of existing GAN-based imputation methods. The prediction task is added into the training stage of GAIN. The generation task and prediction task constitute the two basic tasks of MTGAIN. The prediction task is designed to discover more necessary information about missing values by mining underlying correlations between the generation and prediction tasks. Besides, homoscedastic uncertainty is employed to calculate more reasonable weights for different losses. Experimental results indicate that MTGAIN presents a notable improvement in modeling red tide MTS distribution and imputes missing values more accurately compared with the state-of-the-art methods.

The main contributions of this work are summarized as follows:

(1) We propose a novel multi-task learning-based GAN to exploits the implicit information about missing values contained in the prediction tasks to improve the imputation accuracy for red tide MTS.
(2) To balance the weights between multiple tasks, we utilize the homoscedastic uncertainty to learn the proper allocation of weights. The improvement ensures that the model will not be updated to a fixed direction.
(3) Extensive experiments on a real-world dataset demonstrate that our method achieves state-of-the-art imputation accuracy and model data distribution faithfully, especially in conditions of high missing rates.

The rest of the paper is organized as below. In "Related works", we describe the related works of this paper including the vanilla GAN and homoscedastic uncertainty. The problem formulation is presented in "Problem formulation". Our proposed model MTGAIN and its building block are elaborated in "Proposed method". After extensive experiments in "Experiments", we conclude our work in "Conclusion".

Related works

Vanilla GAN

The vanilla GAN is capable of generating sufficiently realistic data by making the distribution of generated data approximate that of original data when trained with proper strategies. Figure 1 shows the general structure of vanilla GAN which is composed of a generator G and a discriminator D to conduct the competitive process to perform the generation task [20]. G and D are a set of mirrored network structures. G takes the latent random noise z which usually obeys the normal distribution as input to perform deconvolution or decoding operation and obtain generated samples $G(z)$. D is designed to take original samples x and $G(z)$ as input to perform convolution or coding operation. D outputs the probability that $G(z)$ conforms the distribution of x to adjust G to generate more authentic samples. Fully connected networks (FCN) are stacked at the end of both G and D to cope the results of deconvolution and convolution to output the desired forms [19].

G decodes the input noise z to get the generated data as real as possible to confuse D so that D will judge the generated

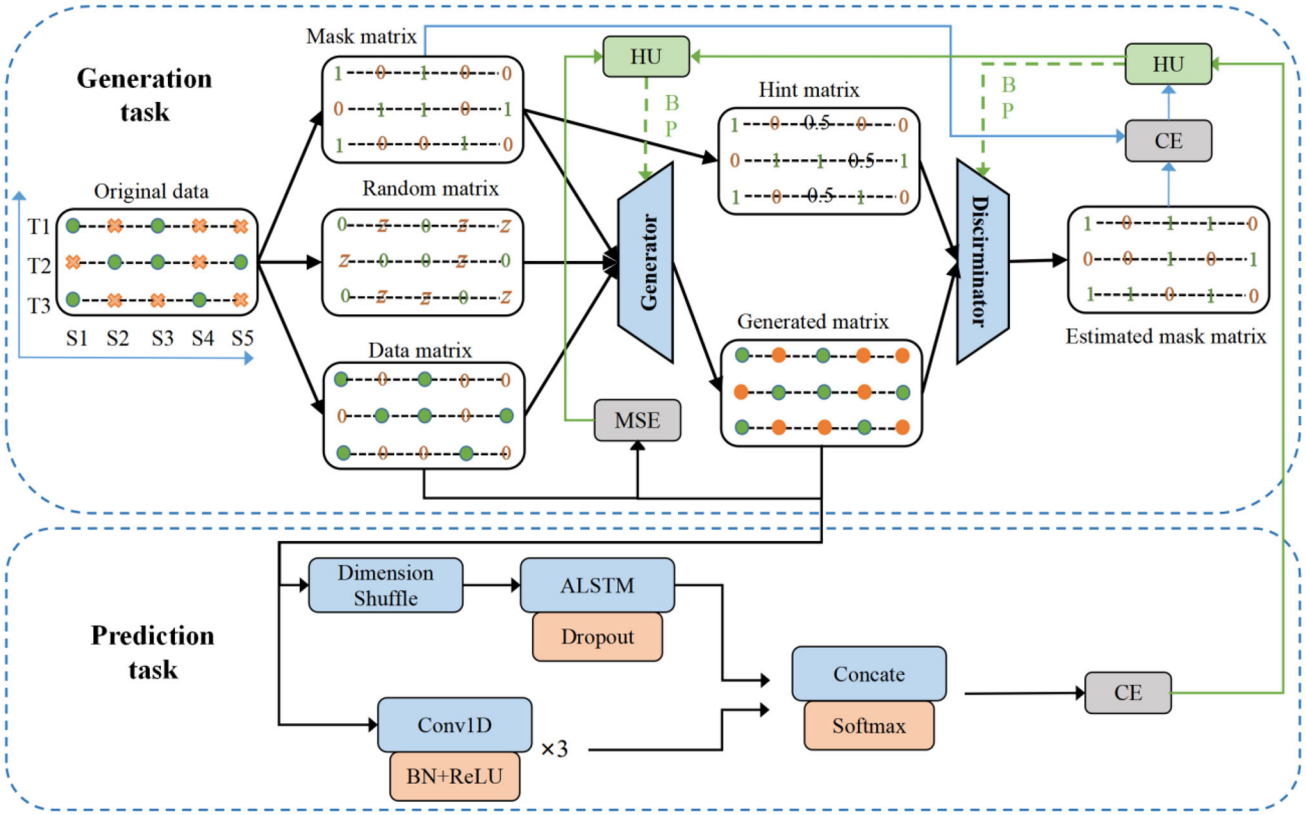

Fig. 2 The architecture of MTGAIN

data as positive labels. The generation process of G conforms to the following formula:

$$\min_{G} E_{z\sim P(z)}[1 - log(D(G(z)))]. \tag{1}$$

The goal of D is to distinguish the real data from the generated data, i.e., to judge the real data as positive labels and judge the generated data as negative labels. The discriminant process of D conforms to the following formula:

$$\max_{D} E_{x\sim P(x)}[log(D(x))] + E_{z\sim P(z)}[log(1 - D(G(z)))]. \tag{2}$$

When the above two loss functions converge after a certain number of iterations, GAN can obtain the mapping between the distribution of original data and the input noise to make the generated data approximate to the original data.

Homoscedastic uncertainty for multi-task learning

Multi-task learning is designed to optimize learning efficiency and generalization of multiple related tasks. It is realized mainly through shared representation to complement the domain information learned by different tasks. One of the key factors influencing the performance of the multi-task learning model is the allocation of weights for different losses. The previous approaches mainly utilize a weighted linear sum of the losses corresponding to different tasks to balance multiple losses. The formula is shown below [33]:

$$L_{total} = \sum_{i} w_i L_i, \tag{3}$$

where w_i and L_i denote the weight and corresponding loss, respectively. The weights in previous work are usually uniform which means each task shares the same importance while this approach ignores the fact that the performance of multi-task learning model highly depends on an appropriate combination of weights between multiple losses [33]. To deal with the problem, some works try to tune the weights manually with practical experience to alleviate the sensitivity of models to weights, which is regarded as a time-consuming process and hard to achieve the best combination.

To find the optimal weights for multi-task learning with a more convenient way, lots of methods are proposed including gradient normalization [34], dynamic task prioritization [35] and dynamic weight averaging [36], etc. Homoscedastic uncertainty is among the most popular. It is the aleatoric uncertainty which remains unchanged for data and varies between multiple tasks and thus an appropriate option for multi-task learning [37]. Homoscedastic uncertainty is intro-

duced to improve the general representation of the integral model and performance of individual task instead of simply performing naive weighted linear combination of the losses, which is able to unearth the correlations between multiple tasks. Reference [38] utilizes the below equation as the minimization objective function of a multi-task model through the homoscedastic uncertainty:

$$L(W, \sigma_1, \cdots, \sigma_i) = \sum_i \left(\frac{1}{2\sigma_i^2} L_i(W) + log\sigma_i \right), \tag{4}$$

where $L_i(W)$ denotes the loss functions of different tasks and W is the learnable model parameter. σ_i is the noise scalar and equivalent to adaptive weights of loss functions $L_i(W)$. σ_i can be either fixed or learned. The last item acts as the regulator of weights to suppress the excessive increment or decrement of weights. The homoscedastic uncertainty is inversely proportional to the weight of corresponding task. Large scale σ_i will decreases the contribution of $L_i(W)$ while small scale σ_i will increase its contribution. The core contribution of the objective function is formulating the reasonable representation of multiple losses. The scale of σ_i is constrained by logarithmic expression term in the objective function to punish objective function when weights are set too large or too small.

Problem formulation

Given a d-dimensional space $\mathcal{X} = \mathcal{X}_1 \times ...\mathcal{X}_d$, the MTS observed at $T = (t_1, ..., t_i)$ is denoted by $X = (X_1, ..., X_d) \in \mathbb{R}^{n \times d}$ taking values in $\mathcal{X}$, where X_{t_i} denotes the observation of X at t_i and $X_{t_i}^j$ is the j-th value of X_{t_i}. $P(X)$ denotes the distribution of X. The value of X is either continuous or binary. In the following example of X, "none" means a missing value.

$$X = \begin{bmatrix} 10 & 4 & \text{none} & 3 & \text{none} & 5 \\ \text{none} & 3 & 4 & \text{none} & 2 & \text{none} \\ 15 & 2 & \text{none} & 9 & 6 & \text{none} \end{bmatrix}, T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}. \tag{5}$$

Suppose that $M \in \mathbb{R}^{n \times d}$ is a mask matrix that takes values in $\{0, 1\}^d$. M indicates whether the values of X exist or not by the following formula:

$$M_{t_i}^j = \begin{cases} 1, & if X_{t_i}^j exists \\ 0, & otherwise \end{cases}. \tag{6}$$

The missing rate of X is defined as below:

$$\text{MisssingRate} = \frac{\sum_{i=1}^{t} \sum_{j=1}^{d} (1 - M_{t_i}^j)}{t \times d}. \tag{7}$$

The main target of imputation task is to reconstruct mask matrix M and impute the missing values of X as accurately as possible.

Proposed method

Model architecture

Modeling precise MTS distribution is beneficial for improving the performance of MTS imputation. GAIN [23] is an appropriate choice to handle the imputation task due to the superior performance in modeling distribution. GAIN adopts the FCN in both G and D and replaces the pooling layers with deconvolution and convolution operations. For more details about GAIN please refer to the original GAIN literature. In this section, the ideal of multi-task learning is introduced into GAIN to dig out more implicit correlations between multiple attributions and prediction tasks to impute missing values more precisely. The prediction task usually reserves part of the information about missing values, which can contribute to the imputation task.

As shown in Fig. 2, the generation task and prediction task are two basic stages of MTGAIN. The generation task is exploited to generate synthetic values which is similar to original values to conduct imputation stage. The prediction task exploits the imputed data to conduct the prediction with a pre-trained LSTM-FCN [39]. The model has been proved to be effective for MTS prediction and the structure is the same as that in original literature. The LSTM-FCN is well trained with original complete data, which means the pre-trained model contains rich label information about input values. The pre-trained model is incorporated into GAIN to restrict the generated data to follow the corresponding distribution, that is to obey the prediction result. This constrain along with the discriminative result from D jointly forces G to generate accurate imputed values.

The green dot and orange cross, respectively, denote the missing and observed values of the original data X. Orange dot represents the imputed values in generated matrix. HU in green frame denotes the homoscedastic uncertainty operation to balance the weights of multiple losses. The green dash line indicates the back propagation. Cross entropy (CE) losses, respectively, from the prediction task and generation task are balanced with HU and the balanced loss is back propagated into D. The mean square errors (MSE) from the generation task and previous balanced loss are further balanced with HU and back propagated into G.

In generation task, G and D are two basic parts of MTGAIN and the minimax game between G and D keeps them in contest. G outputs a generated matrix according to the real observation and D aims to identify which values in the matrix are observed or imputed.

G and D are all composed of multiple FCN layers, which is the same as GAIN. Generated MTS matrix and X are fed into D to conduct the assessment of the authenticity of generated data to achieve the CE loss. The training objective of D is to distinguish that which values in the generated matrix are original or imputed and G will be trained to update its parameters to generate more realistic data. Through the adversarial way between G and D,G is able to produce data that are almost identical to the original ones at the end of the training stage.

Model procedure

Let Z be a d-dimensional random matrix and $\widetilde{X}$ be a data matrix which replaces missing values in X with zero and reserves observed values. G takes $\widetilde{X}$,M and Z as input and outputs imputed values $\overline{X}$. The generation process in generation task can be formulated as follows:

$$\begin{aligned}\overline{X} &= G(\widetilde{X}, M, (1-M)\odot Z)\\ \widehat{X} &= M\odot\widetilde{X} + (1-M)\odot\overline{X})\\ H &= B\odot M + 0.5(1-B)),\end{aligned} \tag{8}$$

where $\odot$ denotes element-wise multiplication. $\widehat{X}$ is the complete generated matrix calculated from $\widetilde{X}$ and $\overline{X}$.$\widehat{X}$ is composed of the observed data from $\widetilde{X}$ and imputed data from $\overline{X}$, which means that some components of $\widehat{X}$ are real and some are fake. That is different from a standard GAN where the output of G is either completely real or completely fake.H denotes the hint matrix which is utilized to provide D with partial information about M to prevent G from overfitting and repeatedly generating several optimal distribution. The hint mechanism guarantees that G can generate desired missing values conditioned by the original incomplete data.$B \in \{0, 1\}^d$ is a random matrix that obeys the following uniform distribution to select elements of M to pass to H:

$$P(B_i^j = b) = \begin{cases} 0.5, & b = 0 \\ 0.5, & b = 1 \end{cases}. \tag{9}$$

G is designed to generate data approximate to the original data. G receives the compressed low-dimensional random noise vector as input. It is trained to learn a mapping from the low-dimensional representation to the original data with no missing values. The generated data from G are regarded as another representation of the original data.D tries to distinguish the real and fake values from the generated matrix by comparing estimated mask matrix $\widehat{M}$ with original mask matrix M. Both G and D utilize FCN layers to map the input matrix into a fixed-dimensional representation.

D is trained to output estimated mask matrix $\widehat{M}$ with regard to the complete generated matrix $\widehat{X}$ and optimize the probability of correctly predicting M. In contrast,G is trained to minimize the possibility of D correctly predicting M. The above procedure can be defined by $V(G, D)$ as follows:

$$V(G, D) = E_{\widehat{X},M,H}[M^T log D(\widehat{X}, H) + (1-M)^T log(1 - D(\widehat{X}, H))]. \tag{10}$$

The objective loss of MTGAIN is a minimax game which is similar to that in the vanilla GAN and follows the formula below:

$$\min_G \max_D V(D, G). \tag{11}$$

According to $V(G, \mathrm{D})$, for the j-th sample $\boldsymbol{m}(j)$ from original data set $\mathbf{m}$ and j-th sample from $\widehat{\boldsymbol{m}}(j)$, the CE loss of these samples is

$$L_D(j) = -\sum_{i=0}[m_i log(\widehat{m}_i) + (1-m_i) log(1-\widehat{m}_i)], \tag{12}$$

where m_i is the i-th element of $\boldsymbol{m}(j)$ and $\widehat{m}_i$ is the i-th element of $\widehat{\boldsymbol{m}}(j)$.$D$ is trained to measure the similarity between $\widehat{M}$ and M by minimizing the following loss L_D:

$$L_D = \sum_{j=1}^{k_1} L_D(j). \tag{13}$$

G is then trained to minimize the weighted sum of the two losses as follows:

$$L_{G1}(j) = -\sum_{i=0}^{d}(1-m_i) log(\widehat{m}_i) \tag{14}$$

$$L_{G2}(j) = \sum_{i=0}^{d} -m_i x_i log(x_i^{'}) \tag{15}$$

$$L_G = \sum_{j=1}^{k_2}(L_{G1}(j) + \alpha L_{G2}(j)). \tag{16}$$

L_{G1} and L_{G2}, respectively, represent the CE and MSE losses and α is a hyper-parameter to measure the proportion between L_{G1} and L_{G2} [23]. L_{G1} is applied to the missing values and L_{G2} is applied to the actually observed values. According to [23], α needs manual adjustment to approach the optimal value. Inspired by the ideal of multi-task learning, the loss of GAIN is extended to a different form. The modified loss of G and D in MTGAIN is shown as below by introducing homoscedastic uncertainty into L_D and L_G:

$$\mathbb{L}_D = \frac{1}{2\sigma_1^2} L_D(W) + \frac{1}{2\sigma_2^2} L_P(W) + log\sigma_1\sigma_2 \tag{17}$$

$$\mathbb{L}_G = \frac{1}{2\sigma_3^2} L_{G1}(W) + \frac{1}{2\sigma_4^2} L_{G2}(W) + log\sigma_3\sigma_4. \tag{18}$$

L_P is the prediction loss achieved by the prediction task. $\mathbb{L}_D$ and $\mathbb{L}_G$, respectively, denote the final form of loss function of D and G in MTGAIN. The training procedure of MTGAIN is shown as the following pseudo code. Firstly, data matrix, random matrix are calculated from original data according to the positions where missing values exist. Then the above matrices are fed into G and generated matrix is obtained as the output of G. The hint matrix is worked out based on mask matrix. Then generated matrix and hint matrix are fed into D and estimated matrix is worked out as the output of D. CE losses L_D and L_{G1} are calculated with mask matrix and estimated mask matrix. MSE loss L_{G2} is calculated with data matrix and generated matrix. The generated matrix is fed into prediction task to achieve prediction loss L_P. Finally, D and G are, respectively, updated with $\mathbb{L}_D$ and $\mathbb{L}_G$ through the back propagation.

Pseudo code of MTGAIN

for number in training epochs **do**
  (1) Training D
  Samples: $\tilde{\mathbf{x}}(j)$,$\mathbf{m}(j)$,$\mathbf{z}(j)$,$\mathbf{b}(j)$ respectively from datasets $\tilde{\mathbf{x}}$,$\mathbf{m}$,$\mathbf{z}$,$\mathbf{b}$
  Ensure: $\mathbf{z}$ and $\mathbf{b}$ respectively follow distribution and $\mathbf{B}$, $j = 1,...,k_1$
  for $j = 1,...,k_1$ **do**
    $\bar{x}(j) \leftarrow G(\tilde{x}(j),m(j),z(j))$
    $\hat{x}(j) \leftarrow m(j) \odot \tilde{x}(j) + (1-m(j)) \odot \bar{x}(j)$
    $h(j) \leftarrow b(j) \odot m(j) + 0.5(1-b(j))$
  end for
  Apply homoscedastic uncertainty to D

$$\mathbb{L}_D = \frac{1}{2\sigma_1^2} L_D(W) + \frac{1}{2\sigma_2^2} L_P(W) + log\sigma_1\sigma_2$$

  Update D with stochastic gradient descent (SGD)

$$\nabla_D \mathbb{L}_D$$

  (2) Training G
  Samples:$\tilde{\mathbf{x}}(j)$,$\mathbf{m}(j)$,$\mathbf{z}(j)$,$\mathbf{b}(j)$ respectively from datasets $\tilde{\mathbf{x}}$,$\mathbf{m}$,$\mathbf{z}$,$\mathbf{b}$
Ensure:$\mathbf{z}$ and$\mathbf{b}$respectively follow distribution $\mathbf{Z}$ and $\mathbf{B}$, $j = 1,...,k_2$
  for $j = 1,...,k_2$ **do**
    $h(j) \leftarrow b(j) \odot m(j) + 0.5(1-b(j))$
  Apply homoscedastic uncertainty to G

$$\mathbb{L}_G = \frac{1}{2\sigma_3^2} L_{G1}(W) + \frac{1}{2\sigma_4^2} L_{G2}(W) + log\sigma_3\sigma_4$$

  Update D with SGD

$$\nabla_G \mathbb{L}_G$$

  end for
end for

Table 1 The AUROC results in various missing rates

Methods	Missing rate (%)						
	10	20	30	40	50	60	70
E2GAN	0.752	0.734	0.713	0.682	0.647	0.614	0.579
GAIN	0.753	0.743	**0.728**	0.679	0.652	0.611	0.568
MTGAIN	**0.763**	**0.746**	0.721	**0.693**	**0.664**	**0.623**	**0.593**

The bold values represent the best performance

Experiments

Dataset description and experiment settings

MTGAIN is evaluated on a real-world red tide MTS dataset and compared with other state-of-the-art methods. The dataset is composed of multiple attributions which influence the occurrence of red tide. Fujian province, located on China southeastern coast, is often plagued by red tides. From 2000 to the middle of 2017, a total of 219 red tides occurred along the coast of Fujian, of which 35 had a huge impact on fishing and aquaculture as well as public health, resulting in a large number of economic losses.

This experiment is conducted on the data collected from the monitoring data of buoys from Dongshan Bay, Fujian Province. The detection time span was from 2007.1 to 2007.3. The monitoring and collection frequency was once every half an hour, which forms a total of 1632 samples. Each sample is labeled with a binary label indicating the occurrence or absence of red tides at the current detection time. The buoy is equipped with multiple marine physical, chemical, biosensors and atmospheric sensors. The dataset uses 8 of them as attributions, respectively: surface temperature (Temp), surface salinity (Salt), saturated oxygen content (SDO), oxygen content (DO), chlorophyll (Chl), turbidity (Turb), pH value (PH), and tide. The models in the experiments are trained to impute missing values and calculate the imputation accuracy as well as post-imputation prediction performance. When imputation operation is accomplished, downstream tasks are performed on the imputed dataset. Excellent imputation methods should have the ability to help downstream models become more effective. Therefore, the prediction task on the red tide dataset is performed to compare the post-imputation prediction performance between different imputation methods directly.

In imputation task, missing values are introduced to the original data in the form of missing completely at random (MCAR) [40], which means the values are removed completely at random and the missing pattern is independent of the observed values. 30%, 50% and 70% of the observed values are discarded by MCAR, respectively. After filling in the missing values with different imputation methods, the imputed dataset is utilized to predict whether red tide will

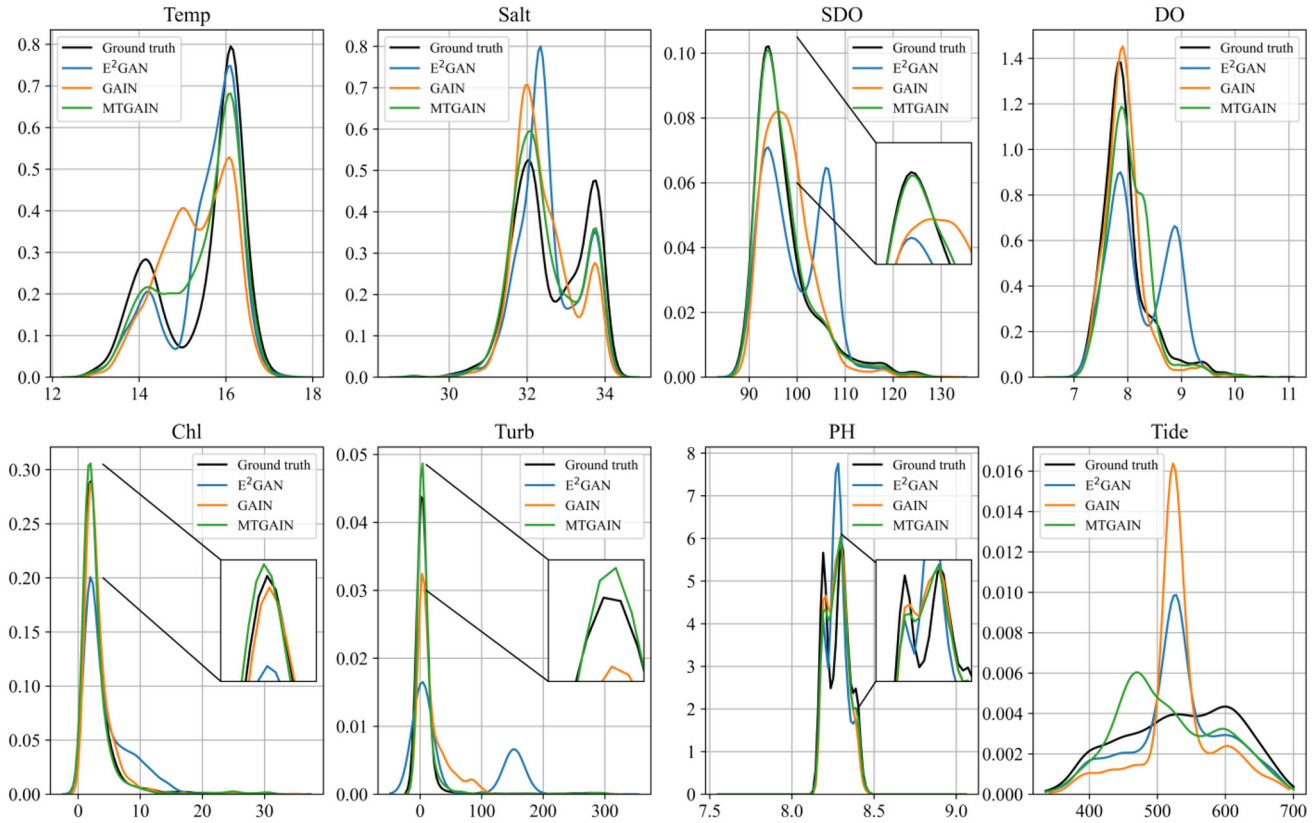

Fig. 3 The distributions of different attributions reconstructed by different methods under 50% missing rate

occur at the current detection time. Each experiment is conducted ten times and 7-cross validation is utilized within each experiment. All experiments of the work were performed with a GPU Nvidia 3060Ti, CPU Intel i7-11700 k and 32 GB of RAM. The software includes Pytorch 1.7.1 + cu110 and Python 3.6.4 in Windows 10 system.

Evaluation metrics and baseline methods

The imputation performance of various imputation methods are quantitatively evaluated with four different evaluation metrics: kernel density estimation (KDE) [41], Pearson correlation coefficient (PCC) [42], root mean square error (RMSE) and area under the receiver operating characteristic curve (AUROC) [21]. See the Appendix for variables and abbreviations. KDE is used to estimate density function of given data and belongs to non-parametric test. The distribution characteristics of every single attribution in the imputed dataset can be seen intuitively through the KDE method. The formula is shown below:

$$\widehat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \tag{19}$$

where x_i denotes the i-th value in the attribution samples and n is the length of x.h denotes the bandwidth which is a smoothing parameter.$K(\cdot)$ represents the kernel function, such as uniform kernel, biweight kernel and Gaussian kernel, etc. The Gaussian kernel is adopted in the experiment as the kernel function of KDE in consideration of the ease in the calculation of waveform synthesis.

PCC is a method to measure the strength of the correlations between two variables and proportional to the strength. The formula is shown below:

$$\rho(X, Y) = \frac{cov(X, Y)}{\sqrt{D(X)}\sqrt{D(Y)}}, \tag{20}$$

where $cov(\cdot)$ denotes the covariance of variable X and Y. $D(\cdot)$ is the variance. The value is closer to 1 when the correlations between the variables become stronger. PCC can be used to check whether the correlations between the imputed attributions are consistent with that of original data. The closer PCC is to that of original data, the better the method can dig out the relationships between different attributions. RMSE of the original and imputed values at the corresponding positions is utilized to compare the imputation performances of different methods directly. Due to the fact that the real-world red tide dataset is imbalanced, which means the occurrence

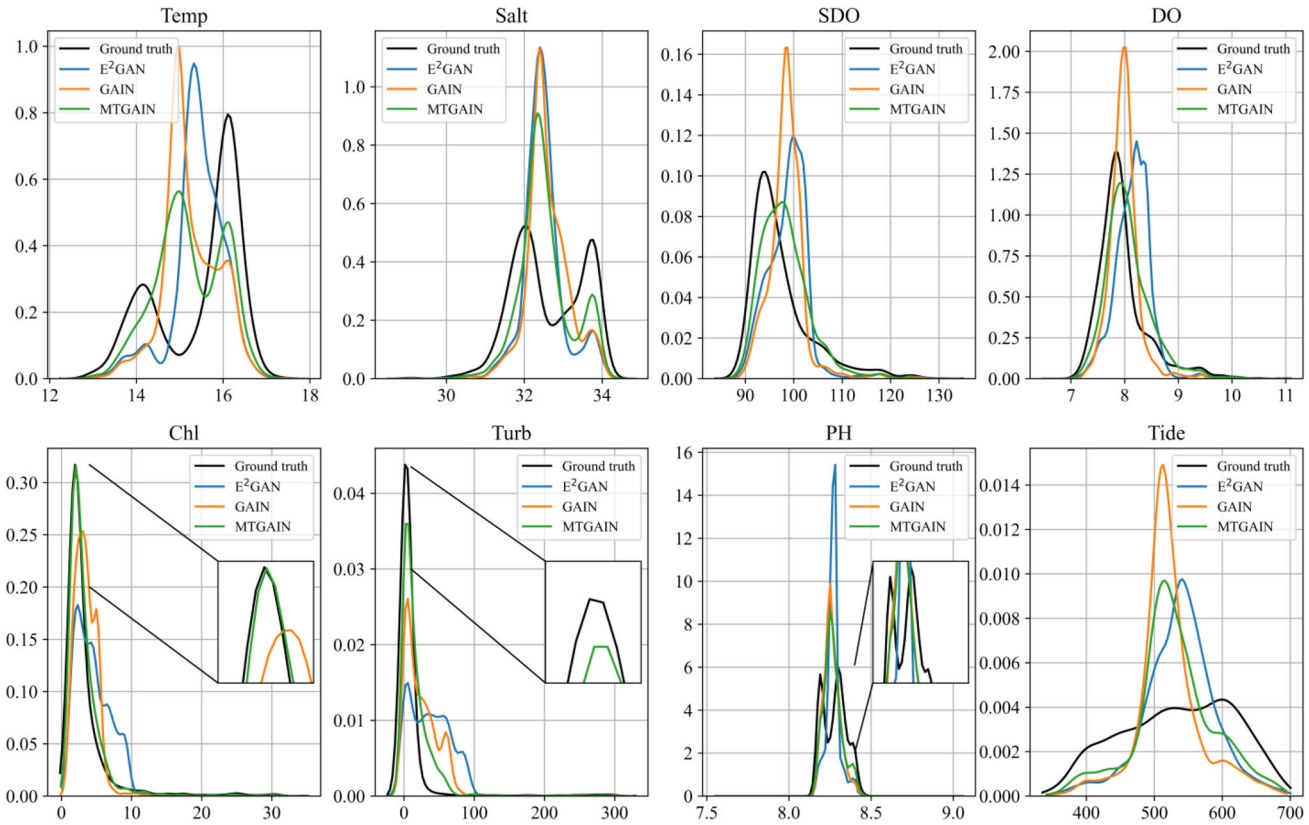

Fig. 4 The distributions of different attributions reconstructed by different methods under 70% missing rate

of red tide is concentrated in a few days or months and the number is small, AUROC is a more appropriate evaluation metric to compare accuracy of post-imputation prediction. The advantage of AUROC is that it is not affected by class imbalance and different sample rates will not affect the evaluation results of AUROC.

To evaluate the imputation performance, MTGAIN is compared with some most commonly used missing values imputation methods: GAIN [23], E^{2}GAN [22], VAE [15], SVD [10], KNN [9].

GAIN: It introduces the hint mechanism to help the generator and discriminator learn the real missing pattern.

E^{2}GAN: It utilizes the decay term to the GRUI and simulate the influence of observed values on missing values with different time interval between them.

VAE: It conducts imputation by constraining the latent space distribution to learn important properties of real data.

SVD: It employs approximate matrix restored from eigenmatrix corresponding to the reserved singular values to impute values iteratively.

KNN: It employs the k-nearest neighbor algorithm to find similar samples with normalized Euclidean distances and impute missing values.

Experiment results

KDE comparison

Note that due to the limitation of article layout, only GAIN, E^{2}GAN and MTGAIN are compared with KDE and PCC performance in 50% and 70% missing rate. All imputation methods mentioned in "Model procedure" are compared with RMSE performance.

Figures 3 and 4 show the KDE performance by multiple imputation methods including E^{2}GAN, GAIN and MTGAIN in different missing rates. Each figure includes reconstructed and original distribution of eight attributions under 50% and 70% missing rate, respectively. To clearly compare the differences between the model performance, a sub-graph describing the peak area is used in the KDE curves of several attributions including SDO, Chl and Turb. The curves with different color represent the distribution of different attributions reconstructed by corresponding methods. Figures 3 and 4 indicate that even though the KDE performance of each algorithm decreases as missing rate increases, MTGAIN consistently outperforms other models in most scenarios.

It is obvious that in some cases the green curve is closer to the black one than others, taking Salt, SDO, Chl and Turb, for example. In cases of above attributions whose dis-

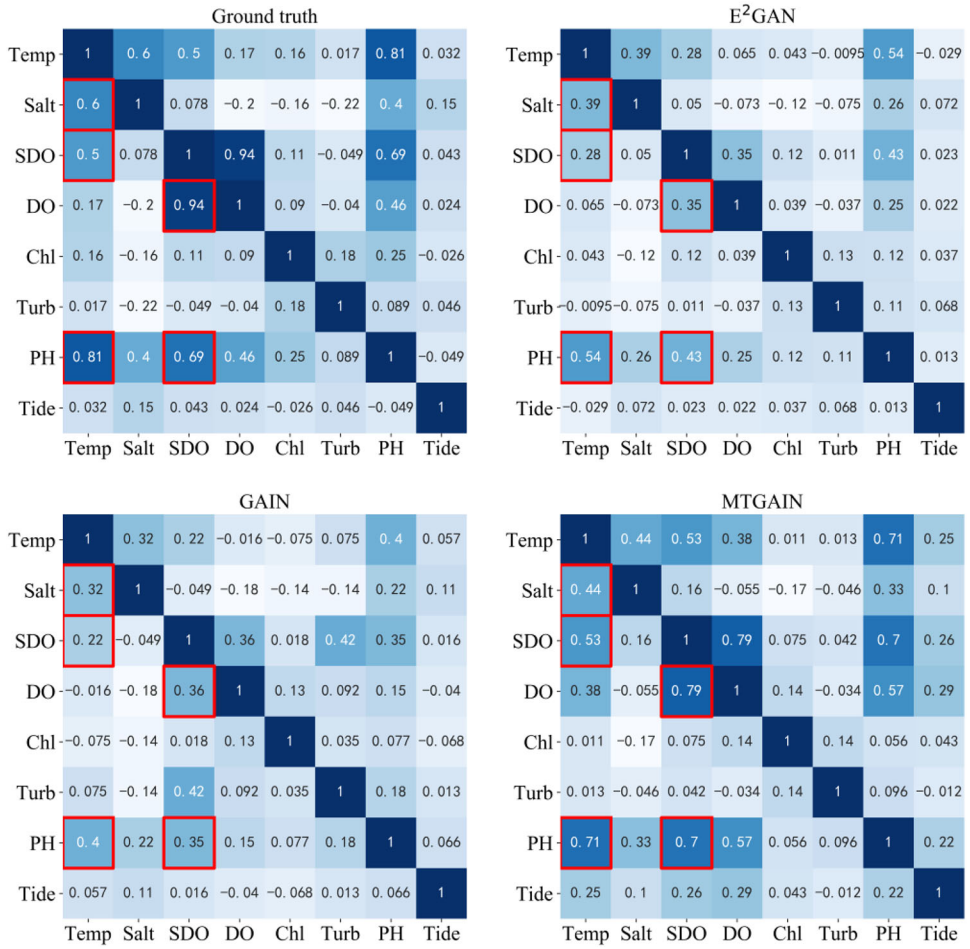

Fig. 5 The PCCs between different attributions reconstructed by different methods under 50% missing rate

tributions appears more complicated with multiple peaks, the curves reconstructed by MTGAIN is more consistent with the original curves than that by other imputation methods. It is probably due to the fact that attributions with complicated distributions produce missing values that are harder to be imputed while MTGAIN is able to capture more implicit information about the missing values from prediction task, thus providing more extra references for imputation. Although in cases of Temp and DO, MTGAIN fails to achieve the best performances, it indeed obtains the second best distribution reconstruction results which are very close to that of GAIN and E^{2}GAN, respectively. While in 70% missing rate, MTGAIN outperforms GAIN and E^{2}GAN in cases of Temp and DO as well as other attributions. It can be seen that the curves reconstructed by MTGAIN at the peak area are closer to the ground truth. In case of Temp in 70% missing rate, although the fitting of MTGAIN at the peak area is not as good as the other two models, MTGAIN successfully simulates the two peaks of the original distribution on the whole, which the other two models fail to achieve. The KDE performance by multiple imputation methods indicates that MTGAIN is superior to other models in modeling complicated distribution, especially in conditions of high missing rate.

PCC performance

Figures 5 and 6 show the PCC performance between different attributions reconstructed by different imputation methods including E^{2}GAN, GAIN and MTGAIN in 50% and 70% missing rates, respectively. The heat map in the upper left corner of each figure is the original PCCs obtained from the original dataset. The values framed in red squares indicate strong positive correlated patterns between corresponding attributions. The larger value means a stronger correlation. The PCCs reconstructed by MTGAIN in general are more consistent with original ones than other methods. The PCCs with red square frames in Fig. 5 reconstructed by MTGAIN are extremely close to ground truth, which indicates great superiority of MTGAIN over other methods in maintaining correlations between attributions. Figure 5 shows that MTGAIN still outperforms other models in many cases despite a decline in performance as miss rate increases.

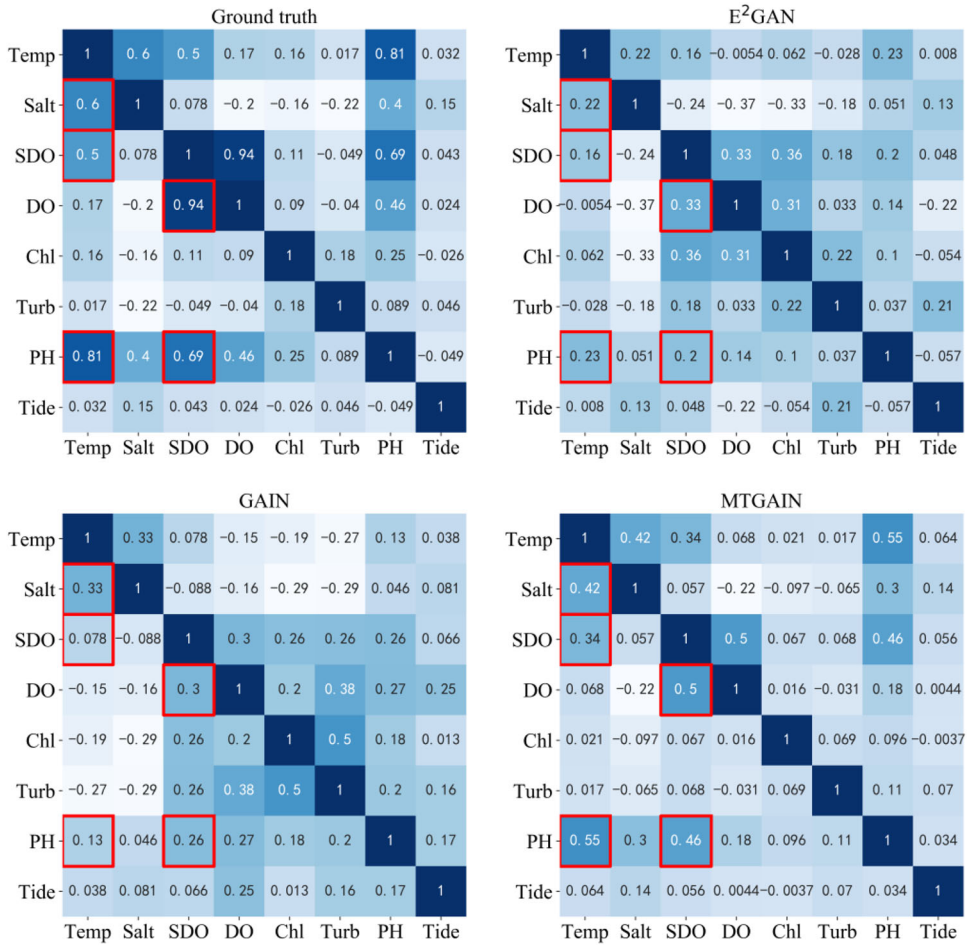

Fig. 6 The PCCs between different attributions reconstructed by different methods under 70% missing rate

In low missing rate, the values imputed by E^{2}GAN and GAIN can still retain the implicit relationships between attributions as much as possible because the correlations are not broken heavily. The performance advantage of MTGAIN is not significantly better than other models in some cases such as PCC between Salt and Temp. In high missing rate, the implicit relationships is destroyed heavily. It is difficult for other methods to capture more information between attributions. MTGAIN adds the loss of prediction task and the pre-trained FCN-LSTM is used to capture the relationships between multiple attributions as much as possible so that the generator and discriminator can be updated in the direction of improving the accuracy of prediction. To improve the accuracy of prediction, the imputed values should conform to the relationships between the original values and the correct prediction results as much as possible, which means they will be generated in the direction of approximating the original data as much as possible. Therefore, adding the prediction loss can improve the performance of imputation and prediction of MTGAIN. These two indicators complement each other. Accurate prediction means that the imputed values are more consistent with the original ones.

On the whole, it can be inferred that as the information contained in the observed data decreases, E^{2}GAN and GAIN fail to capture correlations between attributions and result in poor performance while MTGAIN is still better than others even if missing rate becomes higher. It can be attributed to the characteristics of combination of generation and prediction tasks.

RMSE and AUROC

Figure 7 shows the RMSE results of the five existing approaches and MTGAIN tested when the missing rate varies. As can be seen from Fig. 7, generative models, such as MTGAIN, GAIN, E^{2}GAN and VAE show better RMSE performances when compared with non-generative models including SVD and KNN. MTGAIN achieves the best reconstruction accuracy compared with other approaches in most experiment scenarios. As demonstrated in Fig. 7, MTGAIN is still able to outperform other models and achieve the best imputation results with a relatively tiny increment of RMSE even under high missing rates.

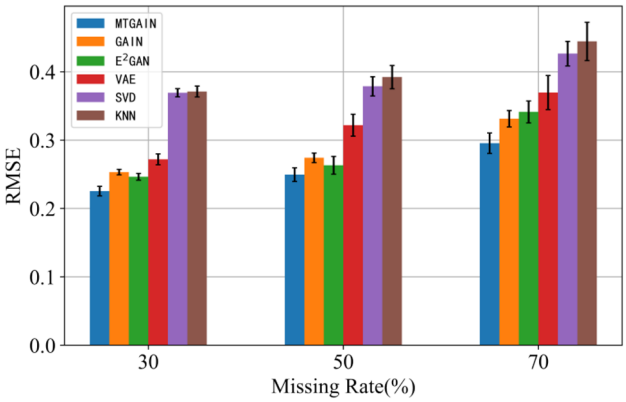

Fig. 7 RMSE performance comparison in MCAR

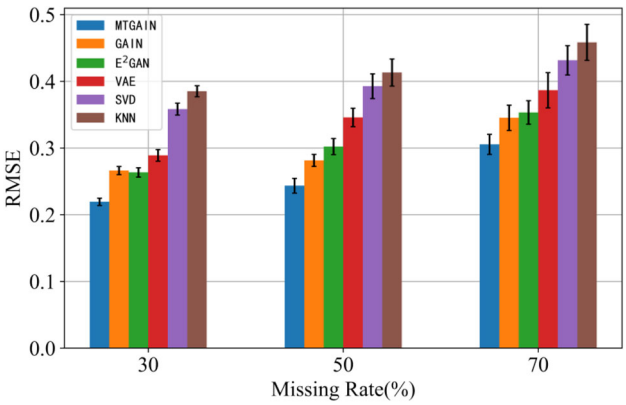

Fig. 8 RMSE comparisons in structural missing

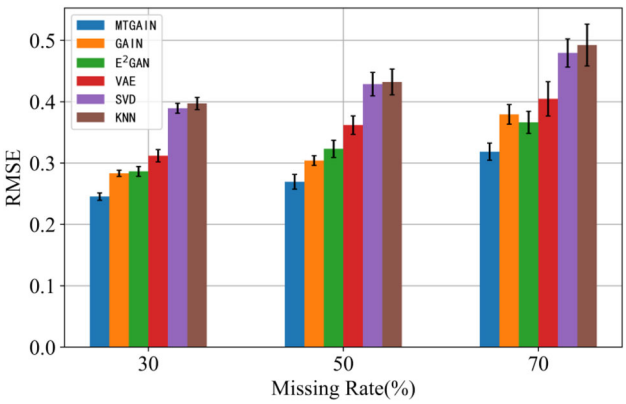

Fig. 9 RMSE comparisons in biased missing

MTGAIN is compared against the same methods with respect to the accuracy of post-imputation prediction when the missing rate varies. For this purpose, AUROC is utilized as the measurement. Table 1 shows MTGAIN yields the best post-imputation prediction performance. In particular, the advantage of MTGAIN over other models enlarges with the increase of miss rate. This is due to the fact that as the information contained in observed data decreases, the implicit information between attributions and prediction tasks extracted by MTGAIN is more conducive to impute missing values, thus improving prediction performance measured by AUROC under high missing rates.

The main reason for MTGAIN to perform well in modeling red tide MTS distribution and further achieve the best imputation performance is that MTGAIN is designed with the purpose of mining latent information of missing patterns by introducing the idea of multi-task learning into GAIN. In this way MTGAIN combines the strong abilities of GAN in modeling distribution with prediction tasks to utilize implicit correlations between MTS attributions.

Robustness analysis

The above results are based on the missing pattern of MACR while it is inevitable to encounter other missing scenarios. To test the robustness of MTGAIN, it is necessary to evaluate the imputation performance in other scenarios. Biased missing and structural missing are two common missing scenarios [28]. In biased missing, 90% of certain attributions are randomly removed. These attributions are regarded as important information which will notably influence the occurrence or absence of red tides. 10% of the remaining attributions are also randomly removed. These attributions are considered less useful which have little impact on red tides. In structural missing, the entire attributions of certain samples are removed. The samples are randomly selected with uniform probability. This missing pattern fits the scenario where the buoy encounters a halt on this day because of power failure or other reasons. In the scenario, no sensors can work normally and the attributions are structurally missed. The RMSE results based on the two missing patterns are shown in Figs. 8 and 9, respectively.

In structural missing, tested models show tiny decline than in MCAR. The main reason is that in structural missing, part of the important attributions are removed and there is no much useful information available. In biased missing, other models present significant decline especially under high missing rates. It is probably due to the fact that with the deletion of almost entire import attributions, these models have no access to useful information to impute missing values and become less reliable. MTGAIN maintains robustness and only exhibits tiny deterioration than in other missing scenarios. This is because it can infer the removed attributions by the prediction task while other models rely heavily on these important attributions. Once the important attributions are removed, other models will show sharp deterioration of performance. The advantage of MTGAIN is that it relies on the prediction task with less emphasis on important attributions, which implies the great applicability to various missing scenarios and the strong robustness of MTGAIN.

Limitations

The main limitations include the expensive cost of computational time and the difficulty to coordinate multiple components during training. On one hand, the inclusion of an extra prediction task increases computational time. In future work, we will investigate the use of model pruning and distillation techniques to alleviate the issue. On the other hand, the appropriate training steps matter for the model to converge. The pre-trained model may be more powerful than other components including the generator and discriminator, which makes it difficult for other components to be updated effectively. It is necessary to try different combinations of training steps for convergence.

Conclusion

In this article, MTGAIN is proposed by combining the generation and prediction tasks for red tide MTS imputation. MTGAIN utilizes multiple complementary tasks to learn a rich representation about original data to unearth the implicit correlations between attributions and prediction tasks. The imputed values maintain the correlations between attributions which are also suitable for prediction tasks. In addition, homoscedastic uncertainty is exploited to balance the weight of losses between generation and prediction tasks to ensure that the parameters of MTGAIN will not be updated to a fixed direction. Experiment results indicate that MTGAIN performs well in modeling distributions of red tide MTS and mining implicit correlations between attributions and prediction tasks, especially under high missing rates. MTGAIN achieves better imputation performance and robustness than the state-of-art models, which makes it a strong alternative to other methods for red tide MTS imputation.

Declarations

Conflict of interest There is no conflict of interest.

Table 2 List of variables and abbreviations mentioned in the article

Variables	Abbreviations	Type	Range
Cross entropy	CE	Continuous	≥ 0
Mean square errors	MSE	Continuous	≥ 0
Surface temperature	Temp	Continuous	12.84–17.36
Surface salinity	Salt	Continuous	29.09–34.26
Saturated oxygen content	SDO	Continuous	88.1–130.6
Oxygen content	DO	Continuous	7.07–10.49
Chlorophyll	Chl	Continuous	0.4–34.6
Turbidity	Turb	Continuous	0.1–306.3
PH value	PH	Continuous	8.15–8.46
Kernel density estimation	KDE	Continuous	≥ 0
Pearson correlation coefficient	PCC	Continuous	0–1
Root mean squard error	RMSE	Continuous	≥ 0
Area under the receiver operating characteristic curve	AUROC	Continuous	0–1

Appendix

See Table 2.

References

1. Xiao S, Yan J, Farajtabar M, Song L, Yang X, Zha H (2019) Learning time series associated event sequences with recurrent point process networks. IEEE Trans Neural Netw Learn Syst 30(10):3124–3136
2. Luan T, Xiaoming L, Jiayu Z, Rong J (2017) Missing modalities imputation via cascaded residual autoencoder. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1405–1414
3. Yi X, Zheng Y, Zhang J, Li T (2016) ST-MVL: filling missing values in geo-sensory time series data. In: International Joint Conference on Artificial Intelligence
4. Serdio F et al (2014) Fault detection in multi-sensor networks based on multivariate time-series models and orthogonal transformations. Inf Fusion 20:272–291
5. Schlomer GL, Bauman S, Card NA (2010) Best practices for missing data management in counseling psychology. J Couns Psychol 57(1):1
6. Bauer S, Schölkopf B, Peters J (2016) The arrow of time in multivariate time series. In: International Conference on Machine Learning pp 2043–2051
7. Lv Y, Duan Y, Kang W, Li Z, Wang F-Y et al (2014) Traffic flow prediction with big data: a deep learning approach. IEEE Trans Intell Transp Syst 16(2):865–873
8. Mehrotra DV, Liu F, Permutt T (2017) Missing data in clinical trials: control-based mean imputation and sensitivity analysis. Pharm Stat 16(5):378–392
9. Zhang S (2012) Nearest neighbor selection for iteratively kNN imputation. J Syst Softw 85(11):2541–2552

10. Yuan X, Han L, Qian S et al (2019) Singular value decomposition based recommendation using imputed data. Knowl-Based Syst 163:485–494
11. García-Laencina PJ, Sancho-Gómez JL, Figueiras-Vidal AR (2010) Pattern classification with missing data: a review. Neural Comput Appl 19(2):263–282
12. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
13. Bengio Y, Gingras F (1996) Recurrent neural networks for missing or asynchronous data. In: Proc. Adv. Neural Inf. Process. Syst
14. Beaulieu-Jones BK, Moore JH (2017) Missing data imputation in the electronic health record using deeply learned autoencoders. In: Pacific Symposium on Biocomputing, pp 207–218
15. Boquet G, Vicario JL, Morell A, et al (2019) Missing data in traffic estimation: A variational autoencoder imputation method. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp 2882–2886
16. Boquet G, Morell A, Serrano J et al (2020) A variational autoencoder solution for road traffic forecasting systems: missing data imputation, dimension reduction, model selection and anomaly detection. Transp Res Part C Emerg Technol 115:102622
17. Nazabal A, Olmos P, Ghahramani Z, Valera I (2020) Handling incomplete heterogeneous data using VAEs. Pattern Recogn 107:107501
18. Goodfellow IJ, et al (2014) Generative adversarial nets. In: Proc Adv Neural Inf. Process Sys
19. Radford A, Metz L, Chintala S (2015) Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434
20. Guo Z, Wan Y, Ye H (2019) A data imputation method for multivariate time series based on generative adversarial network. Neurocomputing 360:185–197
21. Luo Y, et al (2018) Multivariate time series imputation with generative adversarial networks. In: Proc. Adv. Neural Inf. Process. Syst 31
22. Luo Y, Zhang Y, Cai X, et al (2019) E^2GAN: End-to-end generative adversarial network for multivariate time series imputation. In: International Joint Conference on Artificial Intelligence, pp 3094–3100
23. Yoon J, Jordon J, Schaar M (2018) Gain: missing data imputation using generative adversarial nets. In: International Conference on Machine Learning
24. Yang S, Dong M, Wang Y et al (2020) Adversarial recurrent time series imputation. IEEE Trans Neural Netw Learn Syst 91:58560
25. Wang W, Chai Y, Li Y (2022) GAGIN: generative adversarial guider imputation network for missing data. Neural Comput Appl 2:1–14
26. Zhang T, Wang J, Liu J (2021) A gated generative adversarial imputation approach for signalized road networks. IEEE Trans Intell Transp Syst 5:1–17
27. Spinelli I, Scardapane S, Uncini A (2020) Missing data imputation with adversarially-trained graph convolutional networks. Neural Netw 129:249–260
28. Taguchi H, Liu X, Murata T (2021) Graph convolutional networks for graphs containing missing features. Futur Gener Comput Syst 117:155–168
29. Wang P, Zhang T, Zheng Y et al (2022) A multi-view bidirectional spatiotemporal graph network for urban traffic flow imputation. Int J Geogr Inf Sci 63:1–27
30. Le TT, Le Nguyen P, Binh HTT, et al (2021) GCRINT: network traffic imputation using graph convolutional recurrent neural network. IEEE International Conference on Communications, pp 1–6
31. Liang Y, Zhao Z, Sun L (2021) Dynamic spatiotemporal graph convolutional neural networks for traffic data imputation with complex missing patterns. arXiv preprint arXiv:2109.08357
32. Cao W, Wang D, Li J, Zhou H, Li L, Li Y (2018) BRITS: bidirectional recurrent imputation for time series. In: Proc. Adv. Neural Inf. Process.Syst, NIPS 2018
33. Liao Y, Kodagoda S, Wang Y, et al (2016) Understand scene categories by objects: a semantic regularized scene classifier using convolutional neural networks. In: IEEE international conference on robotics and automation, pp 2318–2325
34. Chen Z, Badrinarayanan V, Lee C Y, et al (2018) Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In: International Conference on Machine Learning, pp 794–803
35. Guo M, Haque A, Huang D A, et al (2018) Dynamic task prioritization for multitask learning. In: Proceedings of the European Conference on Computer Vision, pp 270–287
36. Liu S, Johns E, Davison AJ (2019) End-to-end multi-task learning with attention. In: IEEE Conference on Computer Vision and Pattern Recognition, pp 1871–1880
37. Kendall A, Gal Y (2017) What uncertainties do we need in bayesian deep learning for computer vision? arXiv preprint arXiv:1703.04977
38. Kendall A, Gal Y, Cipolla R (2018) Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: IEEE conference on computer vision and pattern recognition, pp 7482–7491
39. Karim F, Majumdar S, Darabi H et al (2017) LSTM fully convolutional networks for time series classification. IEEE Access 6:1662–1669
40. Lee W, Lee S, Byun J et al (2022) Variational cycle-consistent imputation adversarial networks for general missing patterns. Pattern Recogn 56:108720
41. Botev ZI, Grotowski JF, Kroese DP (2010) Kernel density estimation via diffusion. Ann Stat 38(5):2916–2957
42. Benesty J, Chen J, Huang Y (2008) On the importance of the Pearson correlation coefficient in noise reduction. IEEE Trans Audio Speech Lang Process 16(4):57–765