Generative Adversarial Learning for Missing Data Imputation

Xinyang Wang[1†], Hongyu Chen[1†], Jiayu Zhang[1†], Jicong Fan[1,2*]

[1]The Chinese University of Hong Kong, Shenzhen (CUHK-Shenzhen), 518172, China.

[2]Shenzhen Research Institute of Big Data, Shenzhen, 518172, China.

*Corresponding author(s). E-mail(s): fanjicong@cuhk.edu.cn;
Contributing authors: xinyangwang.ryan@gmail.com;
chenhongyu0412@gmail.com; 119020072@link.cuhk.edu.cn;
†These authors contributed equally to this work.

Abstract

Missing data widely exist in industrial problems and lead to difficulties in further modeling and analysis. Recently, a number of deep learning methods have been proposed for missing data imputation and have shown promising performance in various scenarios. Nevertheless, the inputs of imputation networks of these methods are usually incomplete data filled with zeros, which means the missing values always affect the output of the network and the network should be sufficiently large to have a strong denoising ability. Thus, these methods may not provide satisfactory imputation when the missing rate is high. In this work, we present a novel method called GANImputer for missing data imputation. The method is based on the generative adversarial network and the input of the imputation network, namely a generator, is not incomplete data but a low-dimensional latent variable that can be optimized. The optimization process is composed of three stages. First, we optimize a generator via adversarial training. Second, the latent variable is optimized while the generator is fixed. Finally, we fine-tune the generator and latent variable jointly. To analyze the theoretical mechanism of our GANImputer, we provide a generalization error bound with respect to missing not at random, which is practical and meaningful. Our method is tested on five diverse benchmark datasets and the Tennessee Eastman process and outperforms a few deep learning based imputation methods.

Keywords: Missing data imputation, neural networks, deep learning, industrial process.

1 Introduction

Missing data is always a prevalent problem that widely exists in science and engineering because of the cumbersome process of data collection and the inaccessibility of some kinds of data. As a consequence, it is of vital importance to come up with algorithms to handle missing data. In the past decades, many methods have been proposed for handling missing data [1–6]. For instance, Dempster [7] proposed maximum likelihood from incomplete data via the expectation-maximization (EM) algorithm. Van [8] proposed a method called MICE, which does multivariate imputation by chained equations. Stekhoven and Bühlmann [9] presented MissForest by training a random forest on observed data through an iterative imputation scheme. MissForest is particularly useful for categorical data imputation. Low-rank matrix completion (LRMC) algorithms [10–13] impute the missing values by exploiting the potentially low-rank structure of a data matrix and usually have theoretical guarantees for the recovery performance. There are also a few works that focused on recovering the missing values of high-rank matrices of which the data are drawn from a union of subspaces or manifolds [14, 15].

Recently, deep learning models, such as autoencoders [16], have been applied to missing data imputation [17–21]. For instance, Fan [17] proposed to use deep autoencoders to conduct matrix completion. What's more, the well-known generative adversarial network (GAN) [22] has also been adapted to missing data imputation [23, 24]. For instance, to ensure successful operation on incomplete data, GAIN, proposed by [23], feeds additional information, named 'hints', to the discriminator. The 'hints' make the generator generate samples based on the actual underlying data distribution. However, the presence of missing values harms the accuracy of the predicted distribution. As a result, the performance of GAIN may not be satisfactory when the missing rate is high. To handle complex, high-dimensional incomplete data, MisGAN [25] learns a complete data generator along with a mask generator that models the missing data distribution. The missing data is then imputed by equipping the framework with an adversarially trained imputer. Motivated by MisGAN, GAMIN [26] changed the imputation architecture to make the unconditional generator be directly involved in the imputation process and proposed a novel confidence prediction method and a top-k imputation method. Gupta [27] proposed a bi-direction GAN based method for times-series imputation. Kazemi [28] utilized an iterative GAN architecture, which applies the same generator $\mathcal{G}$ with mask $\boldsymbol{M}$ and a shuffled mask $\hat{\boldsymbol{M}}$ to get imputed data $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ respectively. As the input of $\mathcal{G}$ at the second time is $\boldsymbol{x}$, $\boldsymbol{x}$ has more observed values compared to $\hat{\boldsymbol{x}}$. Then, the discriminator $\mathcal{D}$ aims to distinguish between 'imputed' and 're-imputed' data.

It should be pointed out that, in the aforementioned deep learning based missing data imputation methods, the inputs of the imputation networks are incomplete data, which means the performance of the imputation will be directly influenced by the initialization or substitution (e.g. zeros and means) of the missing data, especially when the missing rate is high. Another problem is that, although there have been many deep learning based methods, most missing data imputation methods applied to practical industrial problems are based on shallow models or classical machine learning and optimization methods. For instance, Sahri [29] presented an imputation method based

on the k-nearest neighbor method for the missing values in a dissolved gas analysis dataset. Chen [30] proposed a method based on EM algorithm and Kalman filtering to handle missing data in two-dimensional causal systems. Fan [4] proposed a kernel-based method for industrial missing data imputation. It is worth noting that despite the successful or possibly successful applications of these deep learning imputation methods to industrial problems [23, 25], the theoretical understanding or guarantees are limited.

In this paper, we introduce a novel method, called Generative Adversarial Network Imputer (GANImputer), for missing data imputation. The model is built upon the Generative Adversarial Network (GAN) framework, which includes a discriminator $\mathcal{D}$ and a generator $\mathcal{G}$. Unlike the original GAN, where the discriminator is a binary classifier, the discriminator in GANImputer is formulated as a multi-class classifier, which is trained to maximize the multi-class classification error, while the generator strives to minimize it. After the adversarial training, the generator is fixed and the latent variable $\boldsymbol{Z}$ is optimized. Subsequently, the generator and latent variable are fine-tuned together to impute missing values. We also provide a theoretical analysis of GANImputer's generalization ability, ensuring its effectiveness and reliability. The experimental results on five benchmark datasets with varying missing rates, followed by downstream tasks, demonstrate that GANImputer outperforms strong baselines such as GAIN and MIWAE. Additionally, applying GANImputer to the Tennessee Eastman process further validates our method's effectiveness. Moreover, as GANImputer has three stages, an ablation study was conducted to confirm that each stage of GANImputer significantly contributes to reducing the Root Mean Square Error (RMSE). The contributions of this paper are summarized as follows:

- We propose GANImputer, a novel method for missing data imputation. We elaborate on its model architecture, optimization, and computational complexity.
- We provide generalization error bounds for GANImputer, considering different missing mechanisms.

The remaining content of this paper is organized as follows. Section 2 introduces the related work including a few state-of-the-art missing data imputation methods based on deep learning. The proposed method is illustrated in Section 3. We present the theoretical analysis in Section 4. The numerical results are detailed in Section 5. Finally, Section 6 draws conclusions for this paper.

2 Related work

So far, many deep learning methods have been proposed for missing data imputation [23, 31, 32]. In this section, we discuss the closely related work. In GAIN [23], the generator aims to impute the missing data accurately, and the discriminator tries to distinguish whether a specific element is observed or imputed. A significant improvement from GAN to GAIN is that the output of the discriminator in GAIN is an estimated mask matrix with the same size as the original data instead of a binary value. The GAIN structure also provides a hint matrix to supply additional information about the mask matrix to guarantee that the fake data generated follows the actual distribution.

Another interesting deep imputation algorithm is MIWAE proposed by [33]. It adapts the original importance-weighted autoencoder (IWAE) [34] to a missing data imputation model. MIWAE estimates the log-likelihood by approximating the integrals of marginal probability $\mathcal{P}_\theta(\boldsymbol{x})$ using importance sampling. With the purpose of fitting a deep latent variable model (DLVM) to an incomplete dataset, this algorithm found a sound statistical procedure for training DLVM with missing values by modifying the importance-weighted autoencoder objective, under the assumption that the data are missing at random. Richardson [35] proposed an effective deep imputation method called MCFlow that leverages normalizing flow generative models and Monte Carlo sampling, and addresses the causality dilemma that arises when training models with incomplete data by introducing an iterative learning scheme, which alternately updates the density estimate and the values of the missing entries in the training data. Muzellec [36] provided an optimal transport based missing data imputation method. It is a non-parametric imputation method that does not rely on any parametric assumption for the data and is able to exploit the underlying complex data structure.

3 GANImputer

3.1 Motivation

Let $\boldsymbol{x} = (x_1, ..., x_{d_x})^\top \in \mathbb{R}^{d_x}$ be a (multi-dimension) random variable taking values from real data, $\boldsymbol{m} = (m_1, ..., m_{d_x})^\top \in \{0, 1\}^{d_x}$ be a binary vector, where

$$m_i = \begin{cases} 1, & \text{if } x_i \text{ is observed} \\ 0, & \text{otherwise} \end{cases} \qquad i = 1, ..., d_x. \tag{1}$$

Let $\boldsymbol{X} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_n)^\top \in \mathbb{R}^{n \times d_x}$ and $\boldsymbol{M} = (\boldsymbol{m}_1, ..., \boldsymbol{m}_n)^\top \in \mathbb{R}^{n \times d_x}$. Our goal is to recover the missing values of $\boldsymbol{X}$ from its observed values corresponding to all $m_{ij} = 1$. The key idea of our GANImputer is to learn a latent variable model that can approximate the generating process of $\boldsymbol{X}$, and thus impute the missing values of $\boldsymbol{X}$.

Let $\boldsymbol{z} = (z_1, ..., z_{d_z})^\top \in \mathbb{R}^{d_z}$ be a (multi-dimension) latent variable drawn from some distribution such as $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{d_z})$, where $d_z < d_x$. We want to approximate the generating process of $\boldsymbol{X}$ using a latent variable model $\mathcal{G}$ on $\boldsymbol{z}$, i.e.,

$$\boldsymbol{x}_i \approx \mathcal{G}(\boldsymbol{z}_i), \quad i = 1, \ldots, N, \tag{2}$$

where $\boldsymbol{Z} = (\boldsymbol{z}_1^\top, ..., \boldsymbol{z}_n^\top) \in \mathbb{R}^{d_z \times n}$. Obviously, if both $\mathcal{G}$ and $\boldsymbol{Z}$ are obtained, the missing values of $\boldsymbol{X}$ can be estimated. However, it is difficult to obtain both $\mathcal{G}$ and $\boldsymbol{Z}$ from an incomplete matrix $\boldsymbol{X}$. We learn $\mathcal{G}$ first and then $\boldsymbol{Z}$.

3.2 Adversarial Learning for Generator

For each x_j, we estimate the mean value as μ_i and the variance as σ_j from the observed values, where $j = 1, \ldots, d_x$. Then for $j = 1, \ldots, d_x$, we fill in the missing values of the j-th row of $\boldsymbol{X}$ using random samples drawn from $\mathcal{N}(\mu_j, \sigma_j^2)$ and obtain a full matrix

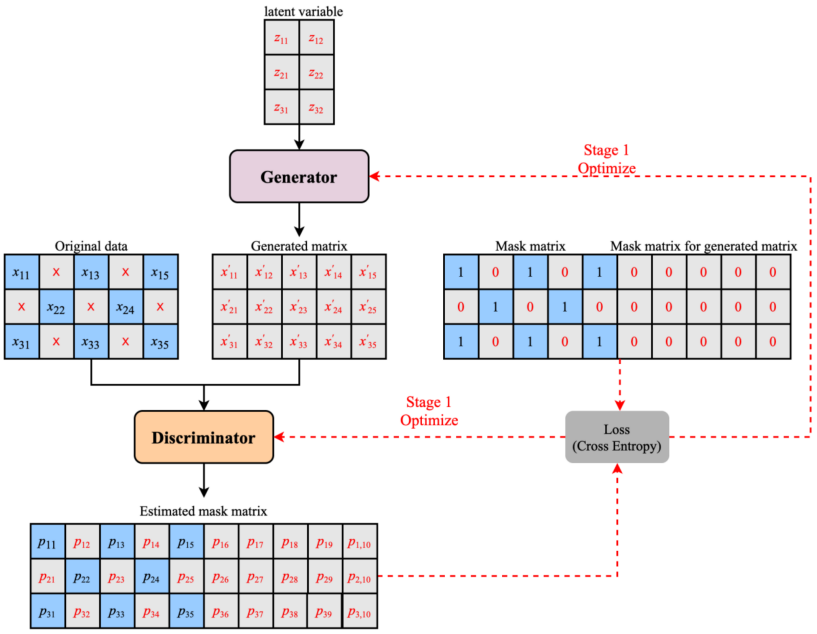


Fig. 1: Stage I of GANImputer. In the example, $d_x = 3$, $d_z = 2$, and $n = 3$. The blue squares correspond to the observed values while the red crosses denote the missing values.

$\bar{\boldsymbol{X}}$, i.e.,

$$\bar{x}_{ij} = \begin{cases} x_{ij}, & \text{if } m_{ij} = 1 \\ \tilde{x}_{ij} \sim \mathcal{N}(\mu_j, \sigma_j^2), & \text{if } m_{ij} = 0. \end{cases} \tag{3}$$

Learning $\mathcal{G}$ from $\bar{\boldsymbol{X}}$ is still difficult because $\bar{\boldsymbol{X}}$ contains random noises or in other words fake values (i.e., $\tilde{x}_{ij}$). To solve the problem, we learn a discriminator $\mathcal{D} : \mathbb{R}^{d_x} \to [0,1]^{d_x}$ that is able to identify whether an element in $\bar{\boldsymbol{X}}$ is real or fake:

$$[\mathcal{D}(\bar{\boldsymbol{x}}_i)]_j = \begin{cases} 1, & \text{if } m_{ij} = 1 \\ 0, & \text{if } m_{ij} = 0. \end{cases} \tag{4}$$

Meanwhile, we learn a generator $\mathcal{G} : \mathbb{R}^d \to \mathbb{R}^{d_x}$ generating nearly-real but fake samples $\hat{\boldsymbol{x}} = \mathcal{G}(\boldsymbol{z})$ that can fool the disciminator $\mathcal{D}$, i.e.,

$$[\mathcal{D}(\hat{\boldsymbol{x}})]_j = 1, \quad \forall j = 1, \ldots, d_x. \tag{5}$$

Let $P(\bar{\boldsymbol{x}})$ be the distribution of $\bar{\boldsymbol{x}}$ corrresponding to (3) and let $P(\boldsymbol{z})$ be the prior distribution of the latent variable $\boldsymbol{z}$. The goal of the discriminator $\mathcal{D}$ is to minimize the classification error on the elements of $\bar{\boldsymbol{x}}$ and $\hat{\boldsymbol{x}}$, i.e.,

$$\min_{\mathcal{D}} \ \mathbb{E}_{P(\bar{\boldsymbol{x}})} \left[\ell(\mathcal{D}(\bar{\boldsymbol{x}}), \bar{\boldsymbol{y}})\right] + \mathbb{E}_{P(\boldsymbol{z})} \left[\ell(\mathcal{D}(\mathcal{G}(\boldsymbol{z})), \hat{\boldsymbol{y}})\right] \tag{6}$$

where $\bar{\boldsymbol{y}} = \boldsymbol{m}$, $\hat{\boldsymbol{y}} \equiv \mathbf{0}$, and ℓ denotes a loss function for classification. The goal of the generator $\mathcal{G}$ is to increase the classification error on the produced $\hat{\boldsymbol{x}}$, i.e.,

$$\max_{\mathcal{G}} \ \mathbb{E}_{P(\boldsymbol{z})} \left[\ell(\mathcal{D}(\mathcal{G}(\boldsymbol{z})), \hat{\boldsymbol{y}}) \right]. \tag{7}$$

Combining (6) and (7), we obtain the following adversarial learning problem:

$$\min_{\mathcal{D}} \max_{\mathcal{G}} \ \mathbb{E}_{P(\bar{\boldsymbol{x}})} \left[\ell(\mathcal{D}(\bar{\boldsymbol{x}}), \bar{\boldsymbol{y}}) \right] + \mathbb{E}_{P(\boldsymbol{z})} \left[\ell(\mathcal{D}(\mathcal{G}(\boldsymbol{z})), \hat{\boldsymbol{y}}) \right]. \tag{8}$$

Now let ℓ be the cross-entropy loss, i.e., $\ell(f, y) = -\left(y \log f + (1-y) \log(1-f)\right)$. Then (8) can be formulated as

$$\begin{aligned} \min_{\mathcal{G}} \max_{\mathcal{D}} \ \mathbb{E}_{P(\bar{\boldsymbol{x}})} & \left[\sum_{j:\bar{y}_j=1} \log\left(\mathcal{D}_j(\bar{\boldsymbol{x}})\right) + \sum_{j:\bar{y}_j=0} \log\left(1 - \mathcal{D}_j(\bar{\boldsymbol{x}})\right) \right] \\ & + \mathbb{E}_{P(\boldsymbol{z})} \left[\sum_{j=1}^{d_x} \log\left(1 - \mathcal{D}_j(\mathcal{G}(\boldsymbol{z}))\right) \right], \end{aligned} \tag{9}$$

where $\mathcal{D}_j$ denotes the j-th output of $\mathcal{D}$.

Solving (9) is the first stage of our GANImputer. The principle is visualized in Fig. 1. In general, to solve the min-max optimization, we update $\mathcal{G}$ and $\mathcal{D}$ alternately. We denote the estimated $\mathcal{G}$ given by (9) as $\hat{\mathcal{G}}$.

3.3 Latent Variable Optimization

$\hat{\mathcal{G}}$ cannot be directly used for recovering the missing values of $\boldsymbol{X}$ because the samples generated by $\hat{\mathcal{G}}$ are independent of the samples in $\boldsymbol{X}$. According to (2), we need to find the $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ that correspond to $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. Therefore, we propose to solve the following minimization problem

$$\min_{\boldsymbol{Z}} \ \frac{1}{n} \|(\bar{\boldsymbol{X}} - \hat{\mathcal{G}}(\boldsymbol{Z})) \odot \boldsymbol{M}\|_F^2 \tag{10}$$

where $\odot$ denotes the Hadamard product between matrices and $\|\cdot\|_F$ denotes the Frobenius norm of matrix. Problem (10) can be solved by gradient-based optimization (e.g. stochastic gradient descent) or even quasi-Newton methods. This is the second stage of our GANImputer. Fig. 2 displays the structure of this stage, where we, based on the $\hat{\mathcal{G}}$ learned from Stage I, aim to optimize the latent variables to reconstruct the observed values of $\boldsymbol{X}$.

Once an estimation of $\boldsymbol{Z}$, denoted by $\hat{\boldsymbol{Z}}$, is obtained from (10), the missing values of $\boldsymbol{X}$ can be estimated as

$$\hat{x}_{ij} = \hat{\mathcal{G}}_j(\hat{\boldsymbol{z}}_i), \quad \forall (i,j) : m_{ij} = 0, \tag{11}$$

where $\hat{\mathcal{G}}_j$ denotes the j-th output of $\hat{\mathcal{G}}$. is the optimized latent variable.

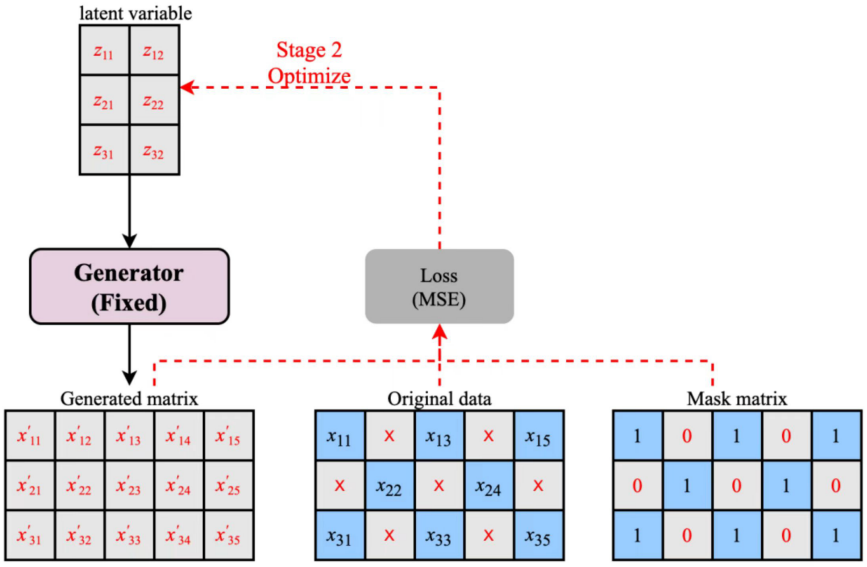


Fig. 2: Stage II of GANImputer. Given $\hat{\mathcal{G}}$ from Stage I, the latent variables $\boldsymbol{Z}$ corresponding to $\boldsymbol{X}$ are optimized.

3.4 Fine-Tuning of Generator and Latent Variable

Note that the $\hat{\mathcal{G}}$ and $\hat{\boldsymbol{Z}}$ given by (9) and (10) respectively may not be optimal or sufficiently accurate because of the nonconvexity of the two optimization problems and the uncertainty given by (3). Therefore, before performing (11), we propose to fine-tune $\hat{\mathcal{G}}$ and $\hat{\boldsymbol{Z}}$ via

$$\min_{\mathcal{G},\boldsymbol{Z}} \frac{1}{n}\|(\bar{\boldsymbol{X}} - \mathcal{G}(\boldsymbol{Z})) \odot \boldsymbol{M}\|_F^2 + \lambda \mathcal{R}(\boldsymbol{Z}, \mathcal{G}), \tag{12}$$

where $\mathcal{R}(\boldsymbol{Z}, \mathcal{G}) = \|\mathbf{Z}\|_F^2 + \sum_{\boldsymbol{W} \in \mathcal{W}} \|\boldsymbol{W}\|_F^2$ denotes the weight-decay or ℓ_2 regularization and $\mathcal{W}$ denotes the set of weight matrices of $\mathcal{G}$. Namely, we tune $\hat{\mathcal{G}}$ and $\hat{\boldsymbol{Z}}$ via performing a few steps of gradient descent for (12). The regularization $\mathcal{R}$ is able to reduce overfitting.

It is worth noting that there is no need to update all the parameters of the generator $\mathcal{G}$. Instead, we propose to fine-tune the last two layers of the generator owing to the flexibility of neural networks. This is the last stage, Stage III, of our GANImputer. Fig. 3 shows the architecture of this stage.

3.5 Overall Algorithm and Computational Complexity

Based on the aforementioned three stages, we present the overall algorithm of GANImputer in Algorithm 1. When the d_x variables of $\boldsymbol{x}$ have different scales, we need to normalize the data before running Algorithm 1. For instance, we can use minimax

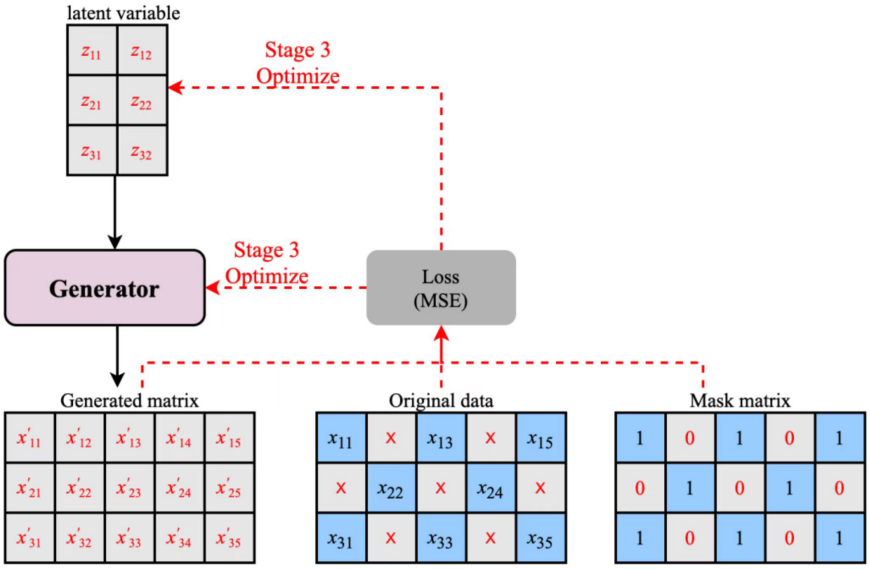


Fig. 3: Stage III of GANImputer. Fine-tune both the Generator $\hat{\mathcal{G}}$ from Stage I and the latent variables $\boldsymbol{Z}$ from Stage II at the same time.

normalization to rescale the samples of each variable to the range of $[0, 1]$. We consider mini-batch optimization. For each epoch, B samples are randomly selected and latent variables are randomly generated from the standard normal distribution. The architecture of the discriminator and generator both follow a shape of low-dimension to high-dimension to low-dimension. In this study, we consider 2 hidden layers in both the discriminator and generator. Particularly, the network structures of the discriminator and generator are $[d_x, 2d_x, 2d_x, d_x]$ and $[d_z, d_x, 2d_x, d_x]$ respectively. Then when optimizing the discriminator, the time complexity and space complexities per iteration are $\mathcal{O}(Bd_x^2)$ and $\mathcal{O}(Bd_x)$ respectively. When optimizing the generator, the time complexity and space complexities per iteration are $\mathcal{O}(Bd_x^2 + Bd_xd_z)$ and $\mathcal{O}(Bd_x + Bd_z)$ respectively.

4 Theoretical Analysis

It is important and necessary to provide theoretical analysis for when and why our GANImputer can predict the missing data successfully. To this end, we analyze the generalization error bound for our GANImputer. Without loss of generality, we consider the following specific structure of the generator:

$$\mathcal{G}(\boldsymbol{z}) = \boldsymbol{W}_L\sigma\left(\boldsymbol{W}_{L-1}\left(\cdots\sigma\left(\boldsymbol{W}_1\boldsymbol{z}\right)\cdots\right)\right) \tag{13}$$

where L is the number of layers, σ denotes the activation functions, and $\boldsymbol{W}_l \in \mathbb{R}^{d_{l+1}d_l}, l \in [L]$. This structure means $d_1 = d_z$ and $d_{L+1} = d_x$. We assume that

Algorithm 1 GANImputer

Input: incomplete data matrix $\boldsymbol{X}$, mask matrix $\boldsymbol{M}$
Output: $\hat{\boldsymbol{Z}}$, $\hat{\mathcal{G}}$

Stage I. Discriminator and generator optimization
Initialize $\mathcal{D}$ and $\mathcal{G}$
Generate $\bar{\boldsymbol{X}}$ using (3) on $\boldsymbol{X}$
while training loss has not converged **do**
    Draw B samples $\bar{\boldsymbol{X}}_B \subset \bar{\boldsymbol{X}}$ randomly
    Draw B i.i.d samples $\boldsymbol{Z}_B$ from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{d_z})$
    Update $\mathcal{D}$ and $\mathcal{G}$ using Adam optimizer on (9)
end while
return $\mathcal{G}$
Stage II. Latent space optimization (fixed generator)
while $\bar{\boldsymbol{X}}_B \neq \emptyset$ **do**
    Perform sampling without replacement: $\bar{\boldsymbol{X}}_B \subset \bar{\boldsymbol{X}}$
    Get the corresponding $\boldsymbol{M}_B \subset \boldsymbol{M}$
    Draw B i.i.d samples $\boldsymbol{Z}_B$ from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{d_z})$
    Update $\boldsymbol{Z}_B$ using Adam optimizer on (10)
end while
return $\boldsymbol{Z}$ (formed by all $\boldsymbol{Z}_B$)
Stage III. Fine-tuning
while training loss has not converged **do**
    Draw B samples $\bar{\boldsymbol{X}}_B \subset \bar{\boldsymbol{X}}$
    Get the corresponding $\boldsymbol{M}_B$ and $\boldsymbol{Z}_B$
    Update $\boldsymbol{Z}_B$ and $\mathcal{G}$ using Adam optimizer on (12)
end while
return $\boldsymbol{Z}$, $\mathcal{G}$

$\max(d_1, \ldots, d_{L+1}) \leq \bar{d}$. Suppose $\{\hat{\boldsymbol{W}}_l\}_{l=1}^L$ and $\hat{\boldsymbol{Z}}$ are the estimated weights and latent variables given by Algorithm 1. Define the set of indices of observed elements $\boldsymbol{X}$ as

$$S := \{(i,j) : \forall m_{ij} = 1\}.$$

We have the following generalization error bound.

Theorem 1. *Suppose x_{ij} is observed with probability p_{ij}, $\forall (i,j) \in S$. Let $\hat{\boldsymbol{X}} = \hat{\mathcal{G}}(\hat{\boldsymbol{Z}})$. Suppose the Lipsschitz constant of σ is ρ, $\max(\|\boldsymbol{X}\|_\infty, \|\hat{\boldsymbol{X}}\|_\infty) \leq \xi$, and $\|\hat{\boldsymbol{W}}_l\|_2 \leq a_l$, $\|\hat{\boldsymbol{W}}_l^\top\|_{2,1} \leq a_l'$, $\forall l \in [L]$[1]. Denote $\kappa = \sqrt{\ln 2\bar{d}^2} \left(\sum_{l=1}^L \left(\frac{a_l'}{a_l} \right)^{2/3} \right)^{3/2}$. Then with*

[1] $\|\cdot\|_2$ denotes the spectral norm of matrix, namely, the largest singular value. $\|\cdot\|_{2,1}$ denotes the $\ell_{2,1}$-norm of matrix, namely, the sum of the ℓ_2 norms of columns of a matrix.

probability at least $1-\frac{2}{nd_x}$*, the following inequality holds:*

$$\begin{aligned}&\frac{1}{nd_x}\sum_{(i,j)\in[n]\times[d_x]}(x_{ij}-\hat{x}_{ij})^2-\frac{1}{nd_x}\sum_{(i,j)\in S}p_{ij}^{-1}(x_{ij}-\hat{x}_{ij})^2\\ \leq&\frac{C\xi\kappa\rho^{(L-1)}\|\hat{\boldsymbol{Z}}\|_F\left(\prod_{l=1}^{L_W}a_l\right)\sqrt{\sum_{(i,j)\in S}p_{ij}^{-2}}}{nd_x},\end{aligned}\tag{14}$$

where C *is an absolute constant.*

The theorem shows the average squared recovery error is upper-bounded by the average (weighted) squared training error plus a term related to the model complexity of the generator $\hat{\mathcal{G}}$. The complexity term of $\hat{\mathcal{G}}$ becomes smaller if the depth and width of the neural network are smaller, which however may increase the training error. The complexity is also related to the $L-1$ power of the Lipschitz constant ρ of the activation function, where $\rho=1$ for commonly used activation functions such as ReLU, Sigmoid, and Tanh.

The theorem is for missing not at random and can also provide a result for missing completely at random by letting $p_{ij}=|S|/(nd_x)$, $\forall(i,j)\in S$. Thus, the bound becomes

$$\begin{aligned}&\frac{1}{nd_x}\sum_{(i,j)\in[n]\times[d_x]}(x_{ij}-\hat{x}_{ij})^2-\frac{1}{|S|}\sum_{(i,j)\in S}(x_{ij}-\hat{x}_{ij})^2\\ \leq&\frac{C\xi\kappa\rho^{(L-1)}\|\hat{\boldsymbol{Z}}\|_F\left(\prod_{l=1}^{L_W}a_l\right)}{\sqrt{|S|}},\end{aligned}\tag{15}$$

which directly shows that observing more samples leads to a tighter generalization error bound. In general, the theorem indicates that our GANImputer is guaranteed to recover the missing values of $\boldsymbol{X}$ with high accuracy when the training error is small and the generator $\hat{\mathcal{G}}$ is not too complex.

5 Numerical Results

In this section, we test GANImputer on five benchmark datasets of machine learning and the Tennessee Eastman process [37], a popular benchmark of process control. In the experiments, GANImputer is compared with a few baselines of missing data imputation. The performance of imputation is measured by RMSE (root-mean-square-error):

$$\text{RMSE}=\sqrt{\frac{1}{|\bar{S}|}\sum_{(i,j)\in\bar{S}}(x_{ij}-\hat{x}_{ij})^2},$$

where $\bar{S}=\{(i,j):\forall m_{ij}=0\}$ denotes the set of indices of missing values. We also compare the performance in downstream tasks such as clustering.

5.1 Experiments on UCI and Kaggle Datasets

In this section, we compare our GANImpuer with six state-of-the-art algorithms of missing data imputation, including Matrix [11], DAE [20], missForest [9], GAIN [23], MIWAE [33], and Singhorn [36] on five benchmark datasets shown below. The first four are from the UCI data repository[2] and the last one is from Kaggle[3]. The details are as follows.

- Bean: 13611 observations, 21 attributes
- Bitcoin: 4839 observations, 736 attributes
- Call: 7195 observations, 21 attributes
- Group: 24016 observations, 2401 attributes
- Superconduct: 21263 observations, 81 attributes

Ablation study for GANImputer We now compare the influence of the three stages on the imputation performance. We sequentially remove one of the three stages while keeping the rest two unchanged. What's more, we take an experiment that only reserves the third stage, for which we optimize the generator and latent variable jointly. The results are reported in Table 1. We see that every stage in GANImputer is necessary and contributes to the reduction of RMSE.

Table 1: RMSE (average ± std) in the ablation study for GANImputer (the missing rate p is 0.2).

Dataset	Stages I, II, III	Stages I, II	Stages I, III	Stages II, III	Stage III
Call	0.0554±0.0015	0.1326±0.0071	0.0561±0.0013	0.0697±0.0066	0.0953±0.0106
Superconduct	0.0401±0.0012	0.1365±0.0047	0.0473±0.0011	0.0536±0.0035	0.0606±0.0041

Table 2: RMSE (average ± std) of GANImputer and the baselines (the missing rate p is 0.2).

Dataset	Bean	Bitcoin	Call	Group	Superconduct
GANImputer	**0.1108±0.0179**	0.0396±0.0118	0.0554±0.0014	**0.0403±0.0023**	**0.0400±0.0011**
GAIN	0.1788±0.0112	**0.0338±0.0004**	0.0741±0.0023	0.0631±0.0011	0.0561±0.0019
MIWAE	0.1210±0.0270	0.1420±0.0090	**0.0510±0.0012**	0.1410±0.0080	0.0460±0.0011
DAE	0.2156±0.0071	0.1994±0.0008	0.1574±0.0011	0.2689±0.0107	0.2108±0.0006
missForest	0.2311±0.0101	0.2066±0.0208	0.1239±0.0007	0.1876±0.0277	0.1753±0.0326
Matrix	0.2342±0.0413	0.1882±0.0028	0.1322±0.0401	0.2511±0.0351	0.2092±0.0982
Singhorn	0.2062±0.0015	0.0863±0.0244	0.0673±0.0009	0.1485±0.0016	0.0830±0.0004

Qualitative analysis of GANImputer To qualitatively analyze the performance of GANImputer, we compare GANImputer with the six baselines when the

[2] http://archive.ics.uci.edu/
[3] https://www.kaggle.com/datasets/taruntiwarihp/bitcoin

Table 3: RMSE (average ± std) of GANImputer and the baselines (the missing rate p is 0.8).

Dataset	Bean	Bitcoin	Call	Group	Superconduct
GANImputer	**0.2203±0.0096**	**0.0971±0.0211**	0.1477±0.0037	**0.1303±0.0093**	**0.0892±0.0010**
GAIN	0.2355±0.0210	0.3287±0.0093	0.1962±0.0198	0.2179±0.0106	0.2466±0.0653
MIWAE	0.2332±0.0280	0.1420±0.0080	**0.0610±0.0009**	0.1490±0.0120	0.0490±0.0011
DAE	0.2881±0.0490	0.2068±0.0008	0.2696±0.0017	0.2853±0.0109	0.2355±0.0011
missForest	0.2367±0.1080	0.2176±0.0028	0.1483±0.0004	0.2148±0.0622	0.1815±0.1091
Matrix	0.2634±0.0006	0.1932±0.0831	0.1502±0.0047	0.2689±0.0360	0.2231±0.0019
Singhorn	0.2282±0.0009	0.1217±0.0173	0.1278±0.0016	0.1492±0.0140	0.1403±0.0006

missing rate (denoted by p) is 0.2 or 0.8. The results are reported in Table 2 and Table 3 respectively. It can be seen that the proposed method GANImputer significantly outperforms other methods in most cases. It should be pointed out that, besides the numerical improvements, our method GANImputer has theoretical guarantees (i.e., Theorem 1), while the baselines are empirical methods and do not have sufficient theoretical guarantees such as generalization bound.

Data visualization We use t-SNE [38] to visualize the dataset Bean (imputed by mean filling, MIWAE, GAIN, MCFlow, and GANImputer) in 2-dimensional space, where the missing rate is 0.2. From Fig. 4, We can see that the three deep imputation methods outperform the naive mean filling. In addition, the overlap between different clusters given by the proposed GANImputer is slightly less than those given by other methods.

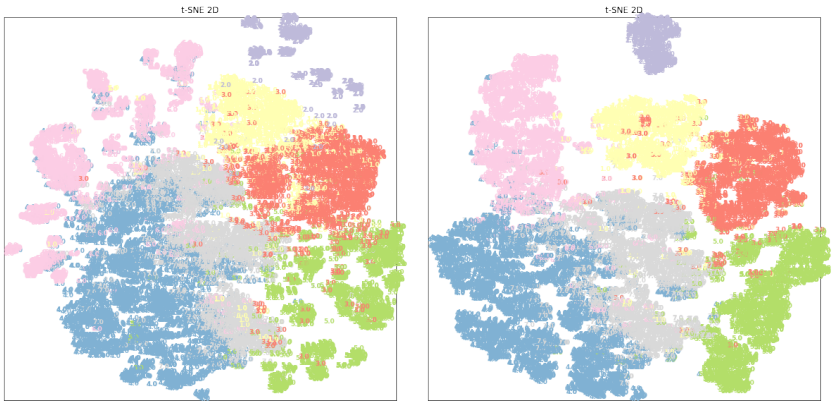
5.2 Experiments on the Tennessee Eastman process

The Tennessee Eastman (TE) process [37], shown by Fig. 5, is a benchmark testbed that has been widely used to test various methods of process control and monitoring [39]. It has 12 manipulated variables, 41 measured variables, and 21 pre-defined faults [37]. In this work, we use the same simulation data generated by [40]. We consider one normal dataset and five fault datasets (Faults 1, 3, 4, 5, and 10), where the number of considered variables is 33 and the number of samples in each dataset is 960.

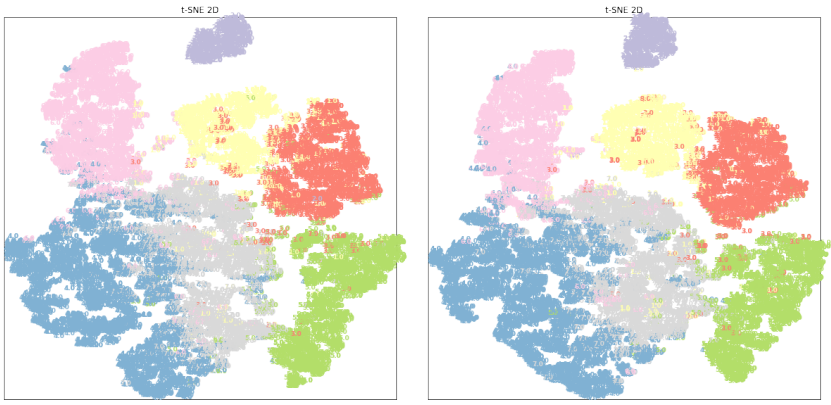
As Table 4 shows, the general performance of GANImputer is better than other missing data imputation methods, especially for F1, F5, and F10 datasets.

5.3 Downstream tasks

To verify whether the imputation process of GANImputer is helpful for our further analysis or not, we perform k-means clustering on Bean and TE data that are imputed by different imputation algorithms. Firstly, Fig. 6 and Fig. 7 illustrate the confusion matrices of the clustering results on dataset Bean imputed by mean and GANImputer when the missing rate is 0.2 or 0.5. We can observe that both the precision and recall of the clustering result on data imputed by GANImputer are higher than those given by mean filling, especially when the missing rate is high. As Table 5 shows,



(a) Mean filling (b) MIWAE



(c) GAIN (d) GANImputer

Fig. 4: Data visualization by t-SNE for data imputed by four imputation algorithms with missing rate 20%.

GANImputer has better performance under both missing rates, which indicates that the dataset imputed by GANImputer demonstrates significantly better performance in downstream tasks such as clustering.

6 Conclusions

We have proposed a novel missing data imputation method based on a deep generative model and provided the generalization error bound. The experimental results showed that the proposed method is more effective than sixe strong baselines of missing data imputation algorithms. One limitation of this work is that we haven't considered the possible dynamics in the data, especially in the TE data. However, it is not difficult

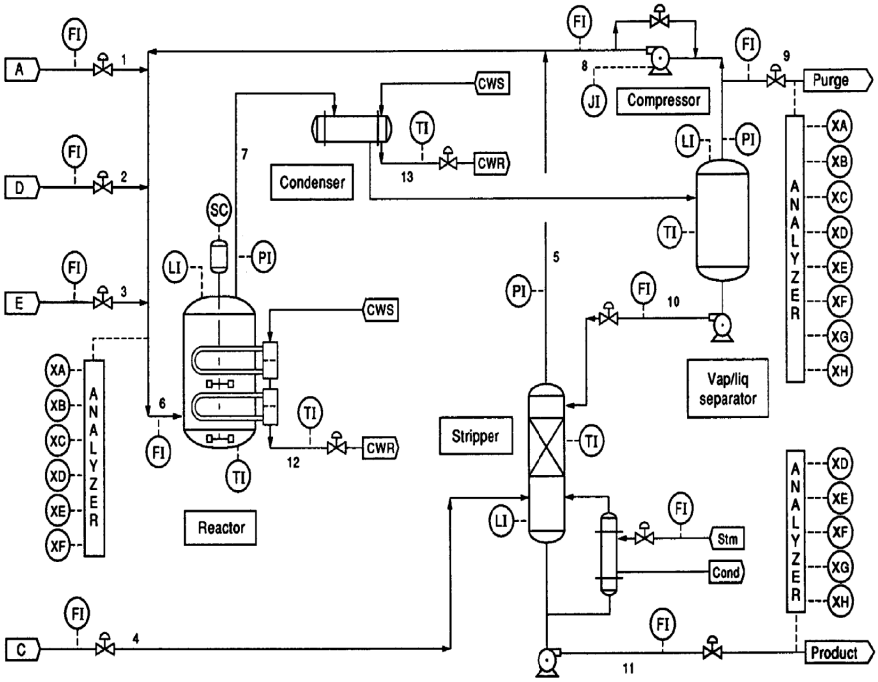
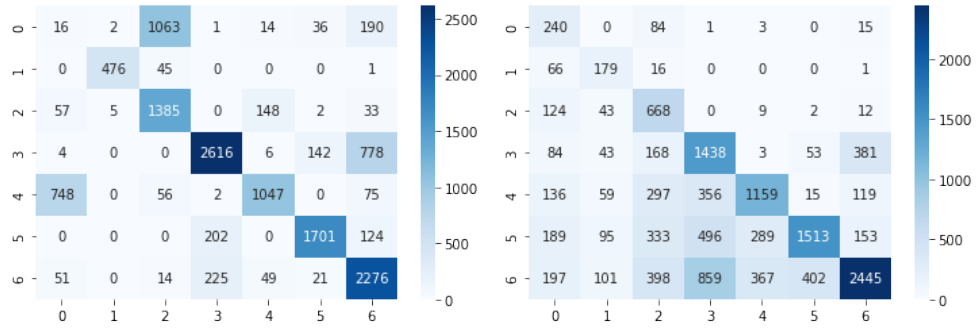


Fig. 5: Tennessee Eastman process [37].



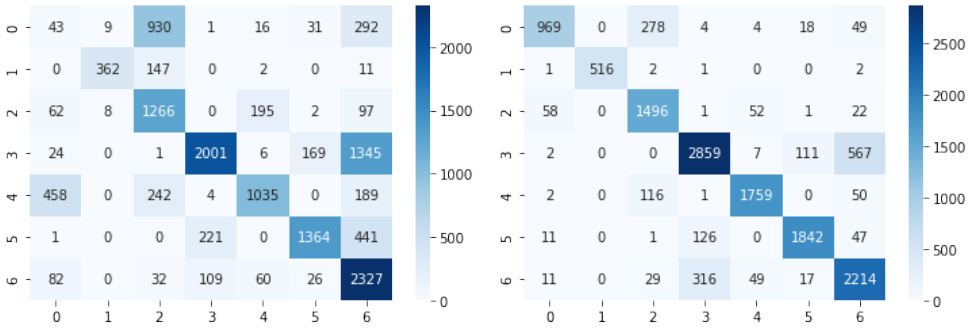
(a) Mean filling (b) GANImputer

Fig. 6: The confusion matrix of clustering on Bean imputed by mean filling and the GANImputer with missing rate 20%.

to extend the proposed method to time series. One can just replace the multilayer perception with a recurrent neural networks.

Table 4: Imputation performance on TE datasset

Dataset	p	GANImputer	DAE	GAIN	MIWAE	Singhorn
Normal	20%	**0.1351**	0.2442	0.1425	0.1592	0.1362
	50%	0.1684	0.2705	0.1668	0.1772	**0.1560**
	80%	0.2144	0.3171	0.2974	0.1993	**0.1940**
F1	20%	**0.1074**	0.1993	0.1255	0.1399	0.1280
	50%	**0.1394**	0.2494	0.1483	0.1682	0.1442
	80%	0.2043	0.2656	0.2754	0.2153	**0.2023**
F3	20%	0.1404	0.1980	**0.1389**	0.1498	0.1492
	50%	0.1689	0.2311	**0.1582**	0.1631	0.1601
	80%	0.2321	0.2644	0.2785	**0.2028**	0.2288
F4	20%	0.1436	0.2302	0.1304	0.1401	**0.1302**
	50%	0.1766	0.2781	**0.1545**	0.1723	0.1633
	80%	**0.2320**	0.3279	0.3184	0.2431	0.2422
F5	20%	**0.1117**	0.2407	0.1258	0.1233	0.1153
	50%	0.1390	0.2964	0.1423	**0.1324**	0.1344
	80%	**0.1841**	0.3016	0.2529	0.1911	0.1921
F10	20%	**0.1323**	0.2171	0.1394	0.1533	0.1444
	50%	**0.1524**	0.2803	0.1569	0.1702	0.1693
	80%	**0.2098**	0.3379	0.2933	0.2328	0.2311



(a) Mean (b) GANImputer

Fig. 7: The confusion matrix of clustering on Bean imputed by mean filling and the GANImputer with missing rate 50%.

Appendix: Proof for Theorem 1

For convenience, we define

$$\begin{cases} \mathcal{L}(\hat{\boldsymbol{X}}) := \frac{1}{nd_x} \sum_{(i,j)\in[n]\times[d_x]} \ell(x_{ij}, \hat{x}_{ij}) \\ \mathcal{L}_S^P(\hat{\boldsymbol{X}}) := \frac{1}{nd_x} \sum_{(i,j)\in S} p_{ij}^{-1} \ell(x_{ij}, \hat{x}_{ij}) \\ \mathcal{L}(\hat{\boldsymbol{X}}') := \frac{1}{nd_x} \sum_{(i,j)\in[n]\times[d_x]} \ell(x_{ij}, \hat{x}'_{ij}) \\ \mathcal{L}_S^P(\hat{\boldsymbol{X}}') := \frac{1}{nd_x} \sum_{(i,j)\in S} p_{ij}^{-1} \ell(x_{ij}, \hat{x}'_{ij}). \end{cases} \tag{16}$$

Table 5: Clustering accuracy on Bean and TE

Dataset	p	GANImputer	GAIN	MIWAE	Singhorn
Bean	20%	**0.8543**	0.7388	0.8443	0.7297
	50%	**0.8563**	0.7104	0.8443	0.8302
TE	20%	**0.3530**	0.3042	0.3211	0.3399
	50%	**0.3399**	0.3044	0.3134	0.3278

We have

$$\begin{aligned}
&|\mathcal{L}(\hat{\boldsymbol{X}}) - \mathcal{L}(\hat{\boldsymbol{X}}')| \\
=&\Big|\tfrac{1}{nd_x} \sum_{(i,j)\in[n]\times[d_x]} \big(\ell(x_{ij},\hat{x}_{ij}) - \ell(x_{ij},\hat{x}'_{ij})\big)\Big| \\
\le&\tfrac{1}{nd_x} \sum_{(i,j)\in[n]\times[d_x]} \big|\ell(x_{ij},\hat{x}_{ij}) - \ell(x_{ij},\hat{x}'_{ij})\big| \\
\le&\tfrac{\eta_\ell}{nd_x} \sum_{(i,j)\in[n]\times[d_x]} \big|\hat{x}_{ij} - \hat{x}'_{ij}\big| \\
\le&\tfrac{\eta_\ell}{\sqrt{nd_x}} \|\hat{\boldsymbol{X}} - \hat{\boldsymbol{X}}'\|_F,
\end{aligned} \tag{17}$$

where the second inequality holds due to the Lipschitz continuity of ℓ. Similarly, we can obtain

$$\begin{aligned}
&|\mathcal{L}_S^P(\hat{\boldsymbol{X}}') - \mathcal{L}_S^P(\hat{\boldsymbol{X}})| \\
=&\left|\frac{1}{nd_x} \sum_{(i,j)\in S} p_{ij}^{-1} \big(\ell(x_{ij},\hat{x}'_{ij}) - \ell(x_{ij},\hat{x}_{ij})\big)\right| \\
\le&\frac{1}{nd_x} \sum_{(i,j)\in S} p_{ij}^{-1} \big|\ell(x_{ij},\hat{x}'_{ij}) - \ell(x_{ij},\hat{x}_{ij})\big| \\
\le&\frac{\eta_\ell}{nd_x} \sum_{(i,j)\in S} p_{ij}^{-1} \big|\hat{x}'_{ij} - \hat{x}_{ij}\big| \\
\le&\frac{\eta_\ell}{nd_x} \sqrt{\Big(\sum_{(i,j)\in S} p_{ij}^{-2}\Big)\Big(\sum_{(i,j)\in S} \big|\hat{x}'_{ij} - \hat{X}_{ij}\big|^2\Big)} \\
\le&\frac{\eta_\ell \sqrt{\sum_{(i,j)\in S} p_{ij}^{-2}}}{nd_x} \|\hat{\boldsymbol{X}}' - \hat{\boldsymbol{X}}\|_F,
\end{aligned} \tag{18}$$

where the third inequality holds due to the Cauchy–Schwarz inequality.

According to the definitions of $\mathcal{L}_S^P(\hat{\boldsymbol{X}}')$ and $\mathcal{L}(\hat{\boldsymbol{X}}')$, we have $\mathbb{E}_S\left[\mathcal{L}_S^P(\hat{\boldsymbol{X}}')\right] = \mathcal{L}(\hat{\boldsymbol{X}}')$. We let $\left|\frac{|S|}{nd_x p_{ij}} \ell(x_{ij},\hat{x}_{ij})\right| \le \frac{\tau_\ell |S|}{nd_x p_{ij}} \triangleq \tau_{ij}$.

The following lemma shows the Hoeffding inequality of sampling without replacement.

Lemma 1. *Let $\mathcal{X} = (x_1, x_2, \ldots, x_n)$ be a finite population of N points and $X_1, X_2, \ldots, X_n$ be a random sample drawn without replacement from $\mathcal{X}$, where $a_i \le$*

$X_i \le b_i$, $i = 1, 2, \ldots, n$. *Then for all* $\varepsilon \ge 0$,

$$\mathbb{P}\left[\frac{1}{n}\sum_{i=1}^{n} X_i - \mu \ge \varepsilon\right] \le \exp\left(-\frac{2n^2\varepsilon^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right) \tag{19}$$

where $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ *is the mean of* $\mathcal{X}$.

According to Lemma 1, we have

$$\mathbb{P}\left[\left|\mathcal{L}(\hat{\boldsymbol{X}}') - \mathcal{L}_S^P(\hat{\boldsymbol{X}}')\right| \ge \varepsilon\right] \le 2\exp\left(-\frac{|S|^2\varepsilon^2}{2\sum_{(i,j)\in S}\tau_{ij}^2}\right).$$

Using union bound for all $\hat{\boldsymbol{X}}' \in \mathcal{S}'$, we get

$$\mathbb{P}\left[\sup_{\hat{\boldsymbol{X}}'\in\mathcal{S}'}\left|\mathcal{L}(\hat{\boldsymbol{X}}') - \mathcal{L}_S^P(\hat{\boldsymbol{X}}')\right| \ge \varepsilon\right] \le 2|\mathcal{S}'|\exp\left(-\frac{|S|^2\varepsilon^2}{2\sum_{(i,j)\in S}\tau_{ij}^2}\right).$$

Letting $\varepsilon = \sqrt{\frac{2\sum_{(i,j)\in S}\tau_{ij}^2}{|S|^2}\ln(nd_x|\mathcal{S}'|)}$, then with probability at least $1 - \frac{2}{nd_x}$, we have

$$\sup_{\hat{\boldsymbol{X}}'\in\mathcal{S}'}\left|\mathcal{L}(\hat{\boldsymbol{X}}') - \mathcal{L}_S^P(\hat{\boldsymbol{X}}')\right| \le \sqrt{\frac{2\sum_{(i,j)\in S}\tau_{ij}^2}{|S|^2}\ln(nd_x|\mathcal{S}'|)}.$$

Now with probability at least $1 - \frac{2}{nd_x}$, we have

$$\begin{aligned}
&\sup_{\hat{\boldsymbol{X}}\in\mathcal{S}}\left|\mathcal{L}(\hat{\boldsymbol{X}}) - \mathcal{L}_S^P(\hat{\boldsymbol{X}})\right| \\
\le &\sup_{\hat{\boldsymbol{X}}\in\mathcal{S}}\left|\mathcal{L}(\hat{\boldsymbol{X}}) - \mathcal{L}(\hat{\boldsymbol{X}}')\right| + \left|\mathcal{L}(\hat{\boldsymbol{X}}') - \mathcal{L}_S^P(\hat{\boldsymbol{X}}')\right| \\
&\quad + \left|\mathcal{L}_S^P(\hat{\boldsymbol{X}}') - \mathcal{L}_S^P(\hat{\boldsymbol{X}})\right| \\
\le &\sup_{\hat{\boldsymbol{X}}\in\mathcal{S}}\frac{\eta_\ell}{\sqrt{nd_x}}\|\hat{\boldsymbol{X}} - \hat{\boldsymbol{X}}'\|_F + \sqrt{\frac{2\sum_{(i,j)\in S}\tau_{ij}^2}{|S|^2}\ln(nd_x|\mathcal{S}'|)} \\
&+ \sup_{\hat{\boldsymbol{X}}\in\mathcal{S}}\frac{\eta_\ell\sqrt{\sum_{(i,j)\in S}p_{ij}^{-2}}}{nd_x}\|\hat{\boldsymbol{X}}' - \hat{\boldsymbol{X}}\|_F \\
\le &\eta_\ell\epsilon\left(\frac{1}{\sqrt{nd_x}} + \frac{\sqrt{\sum_{(i,j)\in S}p_{ij}^{-2}}}{nd_x}\right) + \frac{\tau_\ell\sqrt{\sum_{(i,j)\in S}p_{ij}^{-2}}}{nd_x}\sqrt{2\ln(nd_x|\mathcal{S}'|)}.
\end{aligned}$$

Lemma 2 (Theorem 3.3 of [41], reformulated)**.** *Let* $\boldsymbol{X} = \boldsymbol{W}_L\sigma\left(\boldsymbol{W}_{L-1}\left(\cdots\sigma\left(\boldsymbol{W}_1\boldsymbol{Z}\right)\cdots\right)\right)$, *where* $\boldsymbol{W}_l \in \mathbb{R}^{d_{l+1}d_l}, l \in [L]$, *and*

$\max(d_1, \ldots, d_{L_W+1}) \leq \bar{d}$. *Denote the Lipschitz constant of* σ *by* ρ. *Define*

$$\mathcal{C} := \Big\{ F_{\mathcal{W}}(\boldsymbol{Z}) : \mathcal{W} = (\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L), \\ \|\boldsymbol{W}_l\|_\sigma \leq a_l, \ \left\|\boldsymbol{W}_l^\top\right\|_{2,1} \leq a_l', \ \forall \in [L] \Big\}$$

Then for any $\epsilon > 0$,

$$\begin{aligned} &\ln \mathcal{N}(\mathcal{C}, \epsilon, \|\cdot\|_F) \\ \leq &\frac{\|\boldsymbol{Z}\|_F^2 \ln 2\bar{d}^2}{\epsilon^2} \Big(\rho^{2(L-1)} \prod_{l=1}^{L} a_l^2 \Big) \times \Big(\sum_{l=1}^{L} \Big(\frac{a_l'}{a_l} \Big)^{2/3} \Big)^3. \end{aligned}$$

According to Lemma 2, we have $\ln|\mathcal{S}'| = \ln \mathcal{N}(\mathcal{H}, \|\cdot\|_F, \epsilon) \leq \frac{v}{\epsilon^2}$, where $v = 4\rho^{2(L_W-1)} \|\tilde{\boldsymbol{Z}}\|_F^2 \ln 2\bar{d}^2 \left(\prod_{l=1}^{L_W} a_l^2 \right) \left(\sum_{l=1}^{L} \left(\frac{a_l'}{a_l} \right)^{2/3} \right)^3$. Then we arrive at

$$\begin{aligned} &\sup_{\hat{\boldsymbol{X}} \in \mathcal{S}} |\mathcal{L}(\hat{\boldsymbol{X}}) - \mathcal{L}_S^P(\hat{\boldsymbol{X}})| \\ \leq &\eta_\ell \epsilon \left(\frac{1}{\sqrt{nd_x}} + \frac{\sqrt{\sum_{(i,j)\in S} p_{ij}^{-2}}}{nd_x} \right) \\ &+ \frac{\tau_\ell \sqrt{\sum_{(i,j)\in S} p_{ij}^{-2}}}{nd_x} \sqrt{2\ln(nd_x) + \frac{2v}{\epsilon^2}} \\ \leq &\eta_\ell \epsilon \left(\frac{1}{\sqrt{nd_x}} + \frac{\sqrt{\sum_{(i,j)\in S} p_{ij}^{-2}}}{nd_x} \right) + \frac{C' \tau_\ell \sqrt{v \sum_{(i,j)\in S} p_{ij}^{-2}}}{\epsilon n d_x} \\ \leq &\tau_\ell \left(\frac{1}{\sqrt{nd_x}} + \frac{\sqrt{\sum_{(i,j)\in S} p_{ij}^{-2}}}{nd_x} \right) + \frac{C' \eta_\ell \sqrt{v \sum_{(i,j)\in S} p_{ij}^{-2}}}{nd_x} \\ \leq &\frac{C'' \eta_\ell \sqrt{v \sum_{(i,j)\in S} p_{ij}^{-2}}}{nd_x}. \end{aligned} \tag{20}$$

In the second inequality, $C' > 1$ is some constant sufficiently large. In the third inequality, we have let $\epsilon = \frac{\tau_\ell}{\eta_\ell}$. In the last inequality, $C'' > 1$ is a sufficiently large constant. In our study, ℓ is the square loss, which means $\eta_\ell \leq 4\max(\|\boldsymbol{X}\|_\infty, \|\hat{\boldsymbol{X}}\|_\infty)$. Now we merge the constant 4 in v and C'' into a new constant C. This finished the proof.

Acknowledgement. This work was supported by Grants No. 62106211 and No. 62376236 of the National Natural Science Foundation of China.

Declarations

Conflict of Interest

The authors declared that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Data Availability

The data used in this paper are all from public databases. We have included the corresponding links or references.

References

[1] Vellido, A.: Missing data imputation through gtm as a mixture of t-distributions. Neural Networks **19**(10), 1624–1635 (2006)

[2] 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)

[3] Silva-Ramírez, E.-L., Pino-Mejías, R., López-Coello, M., Cubiles-de-la-Vega, M.-D.: Missing value imputation on missing completely at random data using multilayer perceptrons. Neural Networks **24**(1), 121–129 (2011)

[4] Fan, J., Chow, T.W.S., Qin, S.J.: Kernel-based statistical process monitoring and fault detection in the presence of missing data. IEEE Transactions on Industrial Informatics **18**(7), 4477–4487 (2022) https://doi.org/10.1109/TII.2021.3119377

[5] Adnan, F.A., Jamaludin, K.R., Wan Muhamad, W.Z.A., Miskon, S.: A review of the current publication trends on missing data imputation over three decades: direction and future research. Neural Computing and Applications **34**(21), 18325–18340 (2022)

[6] Zhang, Y., Zhang, R., Zhao, B.: A systematic review of generative adversarial imputation network in missing data imputation. Neural Computing and Applications **35**(27), 19685–19705 (2023)

[7] Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society: Series B (Methodological) **39**(1), 1–22 (1977)

[8] Van Buuren, S., Groothuis-Oudshoorn, K.: mice: Multivariate imputation by chained equations in r. Journal of statistical software **45**, 1–67 (2011)

[9] Stekhoven, D.J., Bühlmann, P.: Missforest—non-parametric missing value imputation for mixed-type data. Bioinformatics **28**(1), 112–118 (2012)

[10] Candès, E.J., Recht, B.: Exact matrix completion via convex optimization. Foundations of Computational mathematics **9**(6), 717–772 (2009)

[11] Yu, H.-F., Rao, N., Dhillon, I.S.: Temporal regularized matrix factorization for high-dimensional time series prediction. In: Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 29. Curran Associates, Inc., ??? (2016)

[12] Fan, J., Ding, L., Chen, Y., Udell, M.: Factor group-sparse regularization for efficient low-rank matrix recovery. Advances in Neural Information Processing Systems **32** (2019)

[13] Liu, B., Li, Y., Xu, Z.: Manifold regularized matrix completion for multi-label learning with admm. Neural Networks **101**, 57–67 (2018)

[14] Fan, J., Chow, T.W.S.: Sparse subspace clustering for data with missing entries and high-rank matrix completion. Neural Networks **93**, 36–44 (2017)

[15] Fan, J., Zhang, Y., Udell, M.: Polynomial matrix completion for missing data imputation and transductive learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 3842–3849 (2020)

[16] Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)

[17] Fan, J., Chow, T.: Deep learning based matrix completion. Neurocomputing **266**, 540–549 (2017) https://doi.org/10.1016/j.neucom.2017.05.074

[18] Fan, J., Cheng, J.: Matrix completion by deep matrix factorization. Neural Networks **98**, 34–41 (2018)

[19] Gondara, L., Wang, K.: Multiple imputation using deep denoising autoencoders. arXiv preprint arXiv:1705.02737 **280** (2017)

[20] Feng, D., Wang, X., Wang, X., Ding, S., Zhang, H.: Deep convolutional denoising autoencoders with network structure optimization for the high-fidelity attenuation of random gpr noise. Remote Sensing **13**(9) (2021) https://doi.org/10.3390/rs13091761

[21] Awan, S.E., Bennamoun, M., Sohel, F., Sanfilippo, F., Dwivedi, G.: A reinforcement learning-based approach for imputing missing data. Neural Computing and Applications **34**(12), 9701–9716 (2022)

[22] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)

[23] Yoon, J., Jordon, J., Schaar, M.: Gain: Missing data imputation using generative

adversarial nets. In: International Conference on Machine Learning, pp. 5689–5698 (2018). PMLR

[24] Wang, W., Chai, Y., Li, Y.: Gagin: generative adversarial guider imputation network for missing data. Neural Computing and Applications **34**(10), 7597–7610 (2022)

[25] Li, S.C.-X., Jiang, B., Marlin, B.: Misgan: Learning from incomplete data with generative adversarial networks. arXiv preprint arXiv:1902.09599 (2019)

[26] Yoon, S., Sull, S.: Gamin: Generative adversarial multiple imputation network for highly missing data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8456–8464 (2020)

[27] Gupta, M., Beheshti, R.: Time-series imputation and prediction with bi-directional generative adversarial networks. arXiv preprint arXiv:2009.08900 (2020)

[28] Kazemi, A., Meidani, H.: Igani: Iterative generative adversarial networks for imputation with application to traffic data. IEEE Access **9**, 112966–112977 (2021)

[29] Sahri, Z., Yusof, R., Watada, J.: Finnim: Iterative imputation of missing values in dissolved gas analysis dataset. IEEE Transactions on Industrial Informatics **10**(4), 2093–2102 (2014) https://doi.org/10.1109/TII.2014.2350837

[30] Chen, J., Huang, B., Ding, F.: Identification of two-dimensional causal systems with missing output data via expectation–maximization algorithm. IEEE Transactions on Industrial Informatics **17**(8), 5185–5196 (2021) https://doi.org/10.1109/TII.2020.3025581

[31] Spinelli, I., Scardapane, S., Uncini, A.: Missing data imputation with adversarially-trained graph convolutional networks. Neural Networks **129**, 249–260 (2020)

[32] Wang, Y., Li, D., Li, X., Yang, M.: Pc-gain: Pseudo-label conditional generative adversarial imputation networks for incomplete data. Neural Networks **141**, 395–403 (2021)

[33] Mattei, P.-A., Frellsen, J.: Miwae: Deep generative modelling and imputation of incomplete data sets. In: International Conference on Machine Learning, pp. 4413–4423 (2019). PMLR

[34] Burda, Y., Grosse, R., Salakhutdinov, R.: Importance weighted autoencoders. arXiv preprint arXiv:1509.00519 (2015)

[35] Richardson, T.W., Wu, W., Lin, L., Xu, B., Bernal, E.A.: Mcflow: Monte carlo flow models for data imputation. In: Proceedings of the IEEE/CVF Conference

on Computer Vision and Pattern Recognition, pp. 14205–14214 (2020)

[36] Muzellec, B., Josse, J., Boyer, C., Cuturi, M.: Missing data imputation using optimal transport. In: International Conference on Machine Learning, pp. 7130–7140 (2020). PMLR

[37] Downs, J.J., Vogel, E.F.: A plant-wide industrial process control problem. Computers Chemical Engineering **17**(3), 245–255 (1993) https://doi.org/10.1016/0098-1354(93)80018-I . Industrial challenge problems in process control

[38] Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)

[39] Joe Qin, S.: Statistical process monitoring: basics and beyond. Journal of Chemometrics **17**(8-9), 480–502 (2003)

[40] Chiang, L.H., Russell, E.L., Braatz, R.D.: Fault Detection and Diagnosis in Industrial Systems. Springer, ??? (2000)

[41] Bartlett, P.L., Foster, D.J., Telgarsky, M.J.: Spectrally-normalized margin bounds for neural networks. Advances in neural information processing systems **30** (2017)