Multiple Imputation via Generative Adversarial Network for High-dimensional Blockwise Missing Value Problems

Zongyu Dai
Department of AMCS
University of Pennsylvania
Philadelphia, USA
daizy@sas.upenn.edu

Zhiqi Bu
Department of AMCS
University of Pennsylvania
Philadelphia, USA
zbu@sas.upenn.edu

Qi Long
Division of Biostatistics
University of Pennsylvania
Philadelphia, USA
qlong@upenn.edu

***Abstract*—Missing data are present in most real world problems and need careful handling to preserve the prediction accuracy and statistical consistency in the downstream analysis. As the gold standard of handling missing data, multiple imputation (MI) methods are proposed to account for the imputation uncertainty and provide proper statistical inference.**

In this work, we propose Multiple Imputation via Generative Adversarial Network (MI-GAN), a deep learning-based (in specific, a GAN-based) multiple imputation method, that can work under missing at random (MAR) mechanism with theoretical support. MI-GAN leverages recent progress in conditional generative adversarial neural works and shows strong performance matching existing state-of-the-art imputation methods on high-dimensional datasets, in terms of imputation error. In particular, MI-GAN significantly outperforms other imputation methods in the sense of statistical inference and computational speed.

***Index Terms*—GAN, neural network, missing data imputation, multiple imputation, missing at random**

I. Introduction

Missing values are common in almost all real datasets and they have a far-reaching impact on the data analysis. For example, integrated data from multiple sources are often analyzed in areas such as the financial analysis and the biomedical research. Since each source only collects a subset of features for its samples, and different sources may have different subsets of features, the blockwise missing data often arise and pose challenges in the downstream analysis. As a concrete example, consider 4 hospitals that collect the test results related to a certain disease (see Figure 1). While the first hospital can run all the tests for its patients, the second hospital can only run the first 4 tests; the third hospital can only run the first 3 tests and the fourth hospital is capable of running all but the 4th test. After integrating all the patient data across the hospitals, the final dataset contains blockwise missing values.

To deal with this blockwise missing data pattern, it is not sufficient to do the complete case analysis (which discards all samples with missing values and often leads to improper inference and biased findings in the subsequent analysis). Instead, we apply imputation methods to fill in the missing values and conduct inference on the imputed datasets for better accuracy and proper statistical inference. Generally speaking, different imputation methods are proposed to work under different missing mechanisms from which the missing data are generated, which include missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). To be specific, MCAR means that the missing probabilities for each entry (or sample) are the same, independent of the values; MAR means that the missing probabilities depend on the observed values, but not on the missing values; MNAR means that the missing probabilities can depend on both the observed and the missing values. In practice, MCAR is the easiest setting where all methods are supposed to work. MNAR is the most difficult setting where no imputation methods work provably without additional structure assumptions. In this work, we develop MI-GAN, a valid imputation method that works on the MAR (and MCAR) mechanism with the theoretical support.

There are two classes of imputation methods, depending on whether the missing data are imputed for one or multiple times, which are referred to as the single imputation (SI) and the multiple imputation (MI), respectively [13]. Single imputation methods, such as the matrix completion [3], [8], [15], [19], often underestimate the uncertainty of the imputed values and cause bias in the downstream analysis. In comparison, multiple imputation methods, such as MICE [2], [4], [21], [25] and our MI-GAN, can overcome this shortage by adequately accounting for the uncertainty of imputed values through the Rubin's rule [13].

In terms of the learning models, state-of-the-art imputation methods are generally categorized into chained equation-based methods [2], [4], [21], [25], random forest-based methods [20], joint modeling [5], [12], [18], [26], matrix completion [3], [8], [15], [19], and deep learning-based methods [6], [9]–[11], [14], [22]–[24]. Chained equation-based methods including MICE are arguably the most popular imputation methods due to their practically stable imputation performance in the low dimensional setting. Additionally, MICE is generally

This work is partly supported by NIH grant R01GM124111.

regarded as an MAR method despite its lack of theoretical guarantees. However, chained equation-based methods can be extremely time-consuming in the high-dimensional setting and their performance often deteriorates significantly as the feature dimension increases. Similarly, MissForest [20] is a popular random forest-based imputation method that suffers from the same issue as MICE. Joint modeling-based methods often assume data are generated from Gaussian distribution and they usually have solid theoretical guarantees under MAR mechanism. Nevertheless, joint modeling-based methods' performance also deteriorates significantly in high dimension or when their assumptions are violated in practice. Matrix completion methods, such as SoftImpute [15], conduct the single imputation based on the low-rank assumption and hence usually lead to improper inference. Recently, many deep learning-based imputation methods have been proposed, such as GAIN [23] and optimal transport-based methods [16]. Specifically, GAIN is a novel multiple imputation method that does not assume the existence of the complete cases. On one hand, GAIN may empirically work for some datasets under MCAR and MAR mechanisms. On the other hand, GAIN is only theoretically supported under MCAR mechanism, unlike our MI-GAN which is supported under MAR. Optimal transport-based methods including the Sinkhorn and Linear RR (both from [16]) have shown empirical outerperformance over other imputation methods under MCAR, MAR and MNAR mechanism, yet the strong performance no longer holds true in the high dimension.

Our contribution: In this paper, we propose two novel GAN-based multiple imputation methods, namely MI-GAN$_1$ and MI-GAN$_2$, which can work for high dimensional block-wise pattern of missing data with a moderate sample size. We highlight that MI-GAN$_1$ is equipped with theoretical guarantees under the MAR mechanism. Importantly, we further propose MI-GAN$_2$ to boost the empirical performance through an iterative training that leverages all the cases, in constrat to MI-GAN$_1$ which only utilizes the complete cases. Extensive synthetic and real data experiments demonstrate that MI-GANs outerperform other state-of-the-art imputation methods in statistical inference, computational speed, and scalability to high dimension, and perform comparably in terms of imputation error.

II. MI-GAN

We start with a description of multivariate-K pattern missing data. We assume there are n samples/cases, each containing p features/variables which are possibly missing. Throughout this paper, we consider high dimensional settings, i.e., $p > n$. Let matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$ denote the data matrix and $\mathbf{X}_{i,j}$ represents the value of the j-th variable for the i-th sample. Additionally, $\mathbf{X}_{i,:}$ and $\mathbf{X}_{:,j}$ stand for the i-th row vector and the j-th column vector, respectively. It is true that any missing data can be pre-processed and grouped into K patterns $\mathbf{X}_{P_k,:}$ for $k \in [K]$, where the samples within each pattern have the same observed and missing features denoted by the index set $\mathbf{obs}(k)$ and the index set $\mathbf{mis}(k)$, respectively. Here P_k is the *index set* for the rows in $\mathbf{X}$ which belong to the k-th pattern. Without loss of generality, we let $\mathbf{X}_{P_1,:}$ denote the set of complete cases for which all features are observed. Furthermore, we define $\mathbf{X}_{P_{-k},:} = \mathbf{X}\backslash\mathbf{X}_{P_k,:}$ as the complement data matrix for $\mathbf{X}_{P_k,:}$. See the example in Figure 1, where the incomplete matrix contains 7 samples which can be grouped into 4 patterns. The samples in the first pattern are all complete and remaining samples contain missing values. Here the pattern index sets are $P_1 = \{1,2\}$, $P_2 = \{3,4\}$, $P_3 = \{5\}$, $P_4 = \{6,7\}$, and observed feature index sets are $\mathbf{obs}(1) = \{1,2,3,4,5,6\}$, $\mathbf{obs}(2) = \{1,2,3,4\}$, $\mathbf{obs}(3) = \{1,2,3\}$, $\mathbf{obs}(4) = \{1,2,3,5,6\}$.

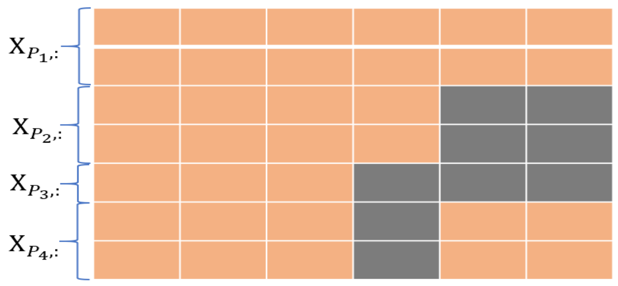

Fig. 1: Multivariate 4-pattern missing data. Orange squares represent observed data and gray squares represent missing data.

Without causing confusion, we let $\mathbf{x}_{\mathbf{obs}(k)}$ and $\mathbf{x}_{\mathbf{mis}(k)}$ denote the observed variables and the missing variables of the k-th pattern. Additionally, we define K *mask vectors* $\mathbf{m}_k \in \mathbb{R}^p$ for each pattern $k \in [K]$: $\mathbf{m}_k(j) = 1$ if $j \in \mathbf{obs}(k)$ otherwise $\mathbf{m}_k(j) = 0$.

A. MI-GAN$_1$:Direct Imputation

Here our goal is to impute the missing values in each pattern. In particular, we aim to generate imputed values from $f(\mathbf{x}_{\mathbf{mis}(k)}|\mathbf{x}_{\mathbf{obs}(k)})$, the conditional distribution of missing variables given observed variables in the k-th pattern. At the high level, MI-GAN$_1$ is an ensemble of $(K-1)$ GANs which are composed of $(K-1)$ pairs of generators and discriminators, and are trained only on complete cases (which belong to the first pattern). Each GAN is used to model one conditional distribution $f(\mathbf{x}_{\mathbf{mis}(k)}|\mathbf{x}_{\mathbf{obs}(k)})$. Figure 2 shows the architecture of our MI-GAN$_1$. The details of MI-GAN$_1$ are described as follows.

Generator G_k: The k-th generator G_k is designed to impute the missing values in the k-th pattern. Let $\mathbf{x}$ denotes a complete case in $\mathbf{X}_{P_1,:}$ and $\mathbf{z}$ denotes an independent p-dimensional noise from $\mathcal{N}(0,\mathbf{I})$. Then $G_k : \mathbb{R}^p \times \mathbb{R}^p \times \{0,1\}^p \to \mathbb{R}^p$ is a function which takes complete case $\mathbf{x}$, noise $\mathbf{z}$ and mask $\mathbf{m}_k$ as input and outputs a vector of imputations. Here the basic idea is to replace the values of $\mathbf{x}$ in the covariate set $\mathbf{mis}(k)$ with a random noise, then feed this partially-true noisy data into the generator to obtain a high-quality imputation. Specifically, for MI-GAN$_1$, the generator G_k entails two steps:

- $\widehat{G}_k(\mathbf{x},\mathbf{z},\mathbf{m}_k) = \widehat{G}_k(\mathbf{x}\odot\mathbf{m}_k + \mathbf{z}\odot(1-\mathbf{m}_k))$ is vector of length p, where $\odot$ denotes the element-wise multiplica-

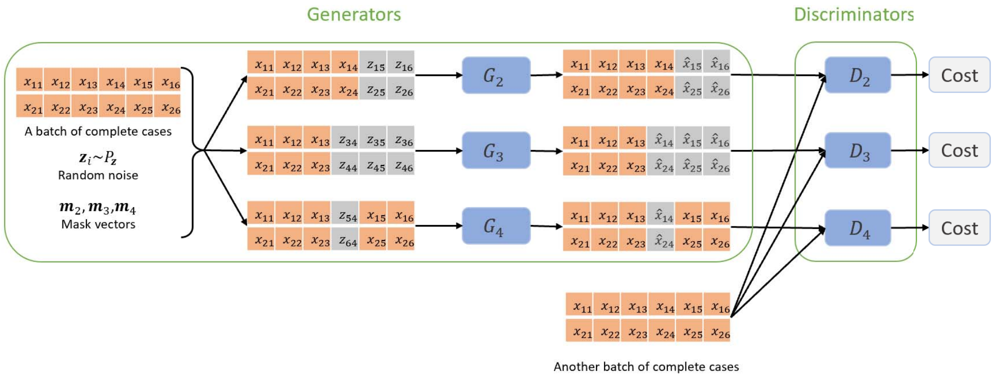
Generators
Discriminators
A batch of complete cases
$\mathbf{z}_i \sim P_{\mathbf{z}}$
Random noise
$\mathbf{m}_2, \mathbf{m}_3, \mathbf{m}_4$
Mask vectors
G_2
G_3
G_4
D_2
D_3
D_4
Cost
Cost
Cost
Another batch of complete cases

Fig. 2: MI-GAN$_1$ applied to the four-pattern missing data in Figure 1

tion and $\widehat{G}_k$ is the generator network that outputs a value for every covariate, even its value was observed.

- Replace $\widehat{G}_k(\mathbf{x} \odot \mathbf{m}_k + \mathbf{z} \odot (1 - \mathbf{m}_k))$ at covariate set $\mathbf{obs}(k)$ with true values to ensure that the observed values are intact and we only impute the missing values. Hence the output of generator is $G_k(\mathbf{x}, \mathbf{z}, \mathbf{m}_k) = \widehat{\mathbf{x}} = \mathbf{x} \odot \mathbf{m}_k + \widehat{G}_k(\mathbf{x}, \mathbf{z}, \mathbf{m}_k) \odot (1 - \mathbf{m}_k)$

Discriminator D_k: Denote the output distribution of k-th generator G_k as P_k and the distribution of complete cases as P. Then discriminator D_k takes $\widehat{\mathbf{x}} \sim P_k$ for $k \in [K]$ and $\mathbf{x} \sim P$ as input. The design of the discriminator depends on the training algorithm used. Here we use Wasserstein GAN [1], [7] framework to train MI-GAN$_1$. Hence the discriminator $D_k : \mathbb{R}^p \to \mathbb{R}$ is a 1-Lipschitz function which estimates the Wasserstein-1 distance between P and P_k using $\widehat{\mathbf{x}}$ and $\mathbf{x}$.

Objective Function: We train the discriminator D_k to estimate the Wasserstein-1 distance (Earth-Mover distance) between P_k and P, and we simultaneously train the generator G_k to minimize this distance. For the k-th pattern, we consider the following objective function,

$$L(D_k, G_k) = \mathbb{E}_{\widehat{\mathbf{x}} \sim P_k}[D_k(\widehat{\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \sim P}[D_k(\mathbf{x})].$$

Then, we ensemble the losses of every GAN with equal weights,

$$L(D_2, G_2, \cdots, D_K, G_K) = \sum_{k=2}^{K} L(D_k, G_k).$$

Hence the objective of MI-GAN$_1$ is the minimax problem given by

$$\min_{G_2, \cdots, G_K} \max_{D_2, \cdots, D_K} L(D_2, G_2, \cdots, D_K, G_K). \qquad (1)$$

B. Theoretical Properties of MI-GAN$_1$

We provide a theoretical analysis of Equation (1) by considering a simplified setting. Let R_k denotes the dummy variable for the k-th pattern ($k \in [K] \backslash \{1\}$). Hence $R_k = 1$ with $R_j = 0$ (for $\forall j \in [K] \backslash \{1\}$ and $j \neq k$) means that this case belongs to the k-th pattern ($k \in [K] \backslash \{1\}$), and $R_k = 0$ (for $\forall k \in [K] \backslash \{1\}$) means this case is complete.

We denote $\mathbf{obs} = \cap_{k=1}^{K} \{\mathbf{obs}(k)\}$ and $\mathbf{mis} = \cup_{k=1}^{K} \{\mathbf{mis}(k)\}$ as the set of commonly observed variables and the set of possibly missing variables across all patterns, respectively, and suppose $\mathbf{obs} \neq \varnothing$. For the example in Figure 1, $\mathbf{obs} = \{1, 2, 3\}$ and $\mathbf{mis} = \{4, 5, 6\}$. Throughout this section, we work with the MAR mechanism such that $\mathbf{x}_{\mathbf{mis}} \perp\!\!\!\perp R_k | \mathbf{x}_{\mathbf{obs}}$ for any $k \in [K] \backslash \{1\}$, which means, given $\mathbf{x}_{\mathbf{obs}}$, $\mathbf{x}_{\mathbf{mis}}$ is independent with R_k.

Theorem 1. *Suppose the generators $G_2, \cdots, G_K$ and discriminators $D_2, \cdots, D_K$ are the optimal solutions of Equation* (1)*, then each generator G_k learns the true conditional distribution $f(\mathbf{x}_{\boldsymbol{mis}(k)} | \mathbf{x}_{\boldsymbol{obs}(k)})$ under our MAR setting.*

Proof of Theorem 1. Let θ_k denote the optimal parameters of the generator G_k and p_{θ_k} denote the probability density of the imputation distribution $\hat{\mathbf{x}}_{\mathbf{mis}(k)}$ conditioned on $\mathbf{x}_{\mathbf{obs}(k)}$. When the optimal discriminator is achieved, it perfectly estimates the Wasserstein-1 distance between the distribution of complete cases and the distribution of G_k's outputs (c.f. [1, Theorem 3]). Hence this Wasserstein-1 distance is zero, which means complete case distribution equals to G_k's output distribution $P = P_k$. Denote ϕ as some probability density derived from the complete case distribution. Then for all (t_1, t_2), the density function of G_k's output distribution P_k is

$$\begin{aligned} &\phi(\mathbf{x}_{\mathbf{obs}(k)} = t_1, \hat{\mathbf{x}}_{\mathbf{mis}(k)} = t_2 | \cap_{k=2}^{K} \{R_k = 0\}) \\ =&\phi(\mathbf{x}_{\mathbf{obs}(k)} = t_1 | \cap_{k=2}^{K} \{R_k = 0\}) p_{\theta_k}(\hat{\mathbf{x}}_{\mathbf{mis}(k)} = t_2 | \mathbf{x}_{\mathbf{obs}(k)} = t_1) \end{aligned}$$

where the second part of last equation comes from G_k's output depends on its input, and $\cap_{k=2}^{K}\{R_k = 0\}$ represents $R_2 = 0, \cdots, R_K = 0$ are all satisfied. For all (t_1, t_2), the density function of complete case distribution P is

$$\begin{aligned} &\phi(\mathbf{x}_{\mathbf{obs}(k)} = t_1, \mathbf{x}_{\mathbf{mis}(k)} = t_2 | \cap_{k=2}^{K} \{R_k = 0\}) \\ =&\phi(\mathbf{x}_{\mathbf{obs}(k)} = t_1 | \cap_{k=2}^{K} \{R_k = 0\}) \phi(\mathbf{x}_{\mathbf{mis}(k)} = t_2 | \mathbf{x}_{\mathbf{obs}(k)} = t_1) \end{aligned}$$

where the last equation comes from the definition of MAR mechanism: $\mathbf{x}_{\mathbf{mis}} \perp\!\!\!\perp R_k | \mathbf{x}_{\mathbf{obs}}$. Therefore, we can conclude

$$p_{\theta_k}(\widehat{\mathbf{x}}_{\mathbf{mis}(k)} = t_2 | \mathbf{x}_{\mathbf{obs}(k)} = t_1) = \phi(\mathbf{x}_{\mathbf{mis}(k)} = t_2 | \mathbf{x}_{\mathbf{obs}(k)} = t_1)$$

which means the optimal G_k learns the true conditional distribution. □

Now we investigate the distribution of imputed samples. by using the optimally trained generator G_k to impute the missing values in the k-th pattern.

Theorem 2. *Suppose the optimal generators $G_2, \cdots, G_K$ and the discriminators $D_2, \cdots, D_K$ are the optimal solutions of Equation* (1)*, then the imputed incomplete cases in the k-th pattern follow the true distribution $f(\mathbf{x}_{obs(k)}, \mathbf{x}_{mis(k)} | R_k = 1)$.*

Proof of Theorem 2. From Theorem 1, we know

$$p_{\theta_k}(\widehat{\mathbf{x}}_{\mathbf{mis}(k)} = t_2 | \mathbf{x}_{\mathbf{obs}(k)} = t_1) = \phi(\mathbf{x}_{\mathbf{mis}(k)} = t_2 | \mathbf{x}_{\mathbf{obs}(k)} = t_1).$$

Hence for all (t_1, t_2),

$$\begin{aligned}
&\phi(\mathbf{x}_{\mathbf{obs}(k)} = t_1, \hat{\mathbf{x}}_{\mathbf{mis}(k)} = t_2 | R_k = 1) \\
=&\phi(\mathbf{x}_{\mathbf{obs}(k)} = t_1 | R_k = 1) p_{\theta_k}(\hat{\mathbf{x}}_{\mathbf{mis}(k)} = t_2 | \mathbf{x}_{\mathbf{obs}(k)} = t_1) \\
=&\phi(\mathbf{x}_{\mathbf{obs}(k)} = t_1 | R_k = 1) \phi(\mathbf{x}_{\mathbf{mis}(k)} = t_2 | \mathbf{x}_{\mathbf{obs}(k)} = t_1) \\
=&\phi(\mathbf{x}_{\mathbf{obs}(k)} = t_1, \mathbf{x}_{\mathbf{mis}(k)} = t_2 | R_k = 1)
\end{aligned}$$

where the first term denotes the density of imputed samples' distribution in the k-th pattern and the last term denotes the density of the k-th pattern sample distribution.

□

C. MI-GAN$_1$ Algorithm

In this section, we provide details of MI-GAN$_1$ algorithm. At the high level, we use an approach similar to that in WGAN with gradient penalty [7], and solve the minimax optimization problem (Equation (1)) in an iterative manner.

When training the GAN for the k-th pattern missing data, we first optimize the discriminator D_k with a fixed generator G_k. Notably, we apply the gradient penalty to the loss function to enforce the Lipschitz constraint [7]:

$$\begin{aligned}
L(D_k) = &\mathbb{E}_{\widehat{\mathbf{x}} \sim P_k}[D_k(\widehat{\mathbf{x}})] - \mathbb{E}_{\mathbf{x} \sim P}[D_k(\mathbf{x})] \\
&+ \lambda_1 \cdot \mathbb{E}_{\widetilde{\mathbf{x}}}[(\|\nabla_{\widetilde{\mathbf{x}}} D(\widetilde{\mathbf{x}})\|_2 - 1)^2]
\end{aligned}$$

where λ_1 is a hyperparameter, $\widetilde{\mathbf{x}} = \epsilon \cdot \mathbf{x} + (1 - \epsilon) \cdot \widehat{\mathbf{x}}$ with $\mathbf{x} \sim P$, $\widehat{\mathbf{x}} \sim P_k$ and $\epsilon \sim U[0, 1]$.

Second, we optimize the generator G_k with newly updated discriminator D_k. Notice that the output of the generator network $\widehat{G}_k(\mathbf{x}, \mathbf{z}, \mathbf{m}_k)$ is a vector of the same length with $\mathbf{x}$. Moreover, MI-GAN$_1$ is trained on complete cases such that $\mathbf{x}$ is fully observed. Thus we add a reconstruction error term to the loss function of G_k to encourage $\widehat{G}_k(\mathbf{x}, \mathbf{z}, \mathbf{m}_k)$ to be close to $\mathbf{x}$. Specifically,

$$\begin{aligned}
L(G_k) = &\mathbb{E}_{\widehat{\mathbf{x}} \sim P_k}[D_k(\widehat{\mathbf{x}})] \\
&+ \lambda_2 \cdot \mathbb{E}_{\mathbf{x} \sim P, \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})}[\|\mathbf{x} - \widehat{G}_k(\mathbf{x}, \mathbf{z}, \mathbf{m}_k)\|_1]
\end{aligned}$$

where λ_2 is a hyperparameter and each element of $\mathbf{z}$ is independently drawn from Gaussian noise $\mathcal{N}(0, 1)$.

After the training process converges, we arrive at the imputation phase. We feed $\mathbf{X}_{P_k,:}$, $\mathbf{m}_k$ and random noise into the well-trained generator G_k to impute the k-pattern missing data. Details are presented in Algorithm 1. When conducting multiple imputation, we run Algorithm 1 for multiple times to account for the uncertainty of parameters in the imputation models (weights and biases of G_k), and combine the imputations with Robin's rule.

Algorithm 1 MI-GAN$_1$: direct imputation

Input: K-pattern missing data $\mathbf{X}$, gradient penalty coefficient λ_1, reconstruction error penalty λ_2, initial parameters for the $K-1$ generators $\theta_{G_2}, \cdots, \theta_{G_K}$, initial parameters for the $K-1$ discriminators (critics) $\theta_{D_2}, \cdots, \theta_{D_K}$, batch size m, number of iterations of the critic per generator iteration n_{critic}, Adam hyperparameters α, β_1, β_2

Output: Imputed matrix

```
1:  while training loss has not converged do
2:      ## Discriminator Optimization
3:      for $j \in \{1, \dots, n_{\texttt{critic}}\}$ do
4:          for $i \in \{1, \dots, m\}$ do
5:              Sample two complete cases $\mathbf{x}, \mathbf{x}'$ from $\mathbf{X}_{P_1,:}$, sample $(K-1)$ noise vectors $\mathbf{z}_k \sim \mathcal{N}(0, \mathbf{I})$ for $k \in [K] \backslash \{1\}$, and a random number $\epsilon \sim U[0, 1]$
6:              for $k \in \{2, \dots, K\}$ do
7:                  $\widehat{\mathbf{x}}_k \leftarrow G_k(\mathbf{x}, \mathbf{z}, \mathbf{m}_k)$
8:                  $\widetilde{\mathbf{x}}_k \leftarrow \epsilon \mathbf{x}' + (1 - \epsilon) \widehat{\mathbf{x}}_k$
9:                  $L_k^{(i)} \leftarrow D_k(\widehat{\mathbf{x}}_k) - D_k(\mathbf{x}') + \lambda_1(\|\nabla_{\widetilde{\mathbf{x}}_k} D(\widetilde{\mathbf{x}}_k)\|_2 - 1)^2$
10:             end for
11:         end for
12:         for $k \in \{2, \dots, K\}$ do
13:             $\theta_{D_k} \leftarrow \text{Adam}(\nabla_{\theta_{D_k}} \frac{1}{m} \sum_{i=1}^m L_k^{(i)}, \theta_{D_k}, \alpha, \beta_1, \beta_2)$
14:         end for
15:     end for
16:     ## Generator Optimization
17:     for $i \in \{1, \dots, m\}$ do
18:         Sample a complete case $\mathbf{x}$ from $\mathbf{X}_{P_1,:}$ and sample $(K-1)$ noise vectors $\mathbf{z}_k \sim \mathcal{N}(0, \mathbf{I})$ for $k \in [K] \backslash \{1\}$
19:         for $k \in \{2, \dots, K\}$ do
20:             $\widehat{\mathbf{x}}_k \leftarrow G_k(\mathbf{x}, \mathbf{z}, \mathbf{m}_k)$
21:             $L_k^{(i)} \leftarrow -D_k(\widehat{\mathbf{x}}_k) - \lambda_2 \|\mathbf{x} - \widehat{G}_k(\mathbf{x}, \mathbf{z}, \mathbf{m}_k)\|_1$
22:         end for
23:     end for
24:     for $k \in \{2, \dots, K\}$ do
25:         $\theta_{G_k} \leftarrow \text{Adam}(\nabla_{\theta_{G_k}} \frac{1}{m} \sum_{i=1}^m L_k^{(i)}, \theta_{G_k}, \alpha, \beta_1, \beta_2)$
26:     end for
27: end while
28: ## Imputation
29: for $k \in \{2, \dots, K\}$ do
30:     for $i \in P_k$ do
31:         Draw a noise vector $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
32:         $\mathbf{X}_{i,:} \leftarrow G_k(\mathbf{X}_{i,:}, \mathbf{z}, \mathbf{m}_k)$
33:     end for
34: end for
35: Output imputed data matrix $\mathbf{X}$.
```

D. MI-GAN$_2$ Algorithm

We notice that Algorithm 1 only exploits the information contained in complete cases to train the model. When the number of complete cases is relatively small, the training of MI-GAN$_1$ is challenging and we may not achieve the optimal generators. Hence the imputation can be negatively affected. To overcome this shortage, we propose an iterative training approach whose model is trained on the whole dataset including incomplete cases. The whole process is presented in Algorithm 2. The iterative training approach requires an initial imputation which can be done by Algorithm 1 (if complete cases exist) or other imputation methods. Similar to MI-GAN$_1$, we create one GAN model for each pattern in MI-GAN$_2$. Then we train a single GAN model and update imputed values for one pattern at each iteration. The newly imputed values are used for the training of the next GAN model in the next iteration. In more details, when updating the imputed values in the k-pattern, we train G_k and D_k on the $\mathbf{X}_{P_{-k},:}$. Then we update $\mathbf{X}_{P_k,:}$ with the newly imputed values from the well-trained generator G_k and this updated $\mathbf{X}_{P_k,:}$ is used to update the $(k+1)$-th pattern.

III. Experiments

In this section, we validate the performance of MI-GANs through extensive synthetic and real data analysis. In all experiments, we not only evaluate the imputation performance but also quantitatively measure the inference ability of MI-GANs with other state-of-the-art imputation methods. Given incomplete dataset, we first conduct SI or MI. Then we fit a linear regression on each imputed dataset and compare the regression coefficient estimate $\hat{\boldsymbol{\beta}}$ for each method (Rubin's rule [13] is used to obtain the final estimate for MI methods). This step is used to evaluate statistical inference performance. Missing data in all experiments are generated from MAR mechanism.

We compare MI-GAN$_1$, MI-GAN$_2$ with 3 benchmarks: **Complete data** analysis, **Complete case** analysis, Column mean imputation (**ColMean Imp**) and 5 other state-of-the-art imputation methods: **MICE** [21], **GAIN** [23], **SoftImpute** [15], **Sinkhorn** [16], and **Linear RR** [16]. Specifically, complete data analysis assume there are no missing values and directly fit a linear regression on the whole dataset. Hence, complete data analysis represent the best result of an imputation method can possibly achieve. Complete case analysis does not conduct imputation and fit a linear regression only using the complete cases. Column mean imputation is feature-wise mean imputation. Here, the complete case analysis and column mean imputation, two naive methods, are used to benchmark potential bias and loss of information under MAR mechanism.

All the experiments run on Google Colab Pro with P100 GPU. For GAIN, Sinkhorn, and Linear RR, we use the open-access implementations provided by their authors, with the default or the recommended hyperparameters in their papers. For SoftImpute, the `lambda` hyperparameter is selected at each run through cross-validation and grid-point search, and we choose `maxit=500` and `thresh=1e-05`.

Algorithm 2 MI-GAN$_2$: iterative imputation

Input: Initial imputation $\mathbf{X}$, imputation times M, burn-in period N, thinning step T, gradient penalty coefficient λ_1, reconstruction error penalty λ_2, initial parameters for the $K-1$ generators $\theta_{G_2}, \cdots, \theta_{G_K}$, initial parameters for the $K-1$ discriminators (critics) $\theta_{D_2}, \cdots, \theta_{D_K}$, batch size m, number of iterations of the critic per generator iteration n_{critic}, Adam hyperparameters α, β_1, β_2

Output: M Imputed matrix

```
1:  for $s \in \{1, \ldots, N + MT\}$ do
2:      for $k \in \{2, \ldots, K\}$ do
3:          while training loss has not converged do
4:              ## Discriminator Optimization
5:              for $t \in \{1, \ldots, n_{\text{critic}}\}$ do
6:                  for $i \in \{1, \ldots, m\}$ do
7:                      Sample two cases $\mathbf{x}, \mathbf{x}'$ from $\mathbf{X}_{P_{-k},:}$, draw a noise vector $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$, and a random number $\epsilon \sim U[0,1]$
8:                      $\widehat{\mathbf{x}}_k \leftarrow G_k(\mathbf{x}, \mathbf{z}, \mathbf{m}_k)$
9:                      $\widetilde{\mathbf{x}}_k \leftarrow \epsilon\mathbf{x}' + (1-\epsilon)\widehat{\mathbf{x}}_k$
10:                     $L_k^{(i)} \leftarrow D_k(\widehat{\mathbf{x}}_k) - D_k(\mathbf{x}') + \lambda_1(\|\nabla_{\widetilde{\mathbf{x}}} D(\widetilde{\mathbf{x}})\|_2 - 1)^2$
11:                 end for
12:                 $\theta_{D_k} \leftarrow \text{Adam}(\nabla_{\theta_{D_k}} \frac{1}{m} \sum_{i=1}^m L_k^{(i)}, \theta_{D_k}, \alpha,$
13: $\beta_1, \beta_2)$
14:             end for
15:             ## Generator Optimization
16:             for $i \in \{1, \ldots, m\}$ do
17:                 Sample a case $\mathbf{x}$ from $\mathbf{X}_{P_{-k},:}$ and draw a noise vector $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
18:                 $\widehat{\mathbf{x}}_k \leftarrow G_k(\mathbf{x}, \mathbf{z}, \mathbf{m}_k)$
19:                 $L_k^{(i)} \leftarrow -D_k(\widehat{\mathbf{x}}_k) - \lambda_2\|\mathbf{x} - \widehat{G}_k(\mathbf{x}, \mathbf{z}, \mathbf{m}_k)\|_1$
20:             end for
21:             $\theta_{G_k} \leftarrow \text{Adam}(\nabla_{\theta_{G_k}} \frac{1}{m} \sum_{i=1}^m L_k^{(i)}, \theta_{G_k}, \alpha, \beta_1, \beta_2)$
22:         end while
23:         ## Imputation
24:         for $i \in P_k$ do
25:             Draw a noise vector $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
26:             $\mathbf{X}_{i,:} \leftarrow G_k(\mathbf{X}_{i,:}, \mathbf{z}, \mathbf{m}_k)$
27:         end for
28:     end for
29:     if $s > N$ and $T \big| (s - N)$ then
30:         output $\mathbf{X}$
31:     end if
32: end for
```

Models	Style	Time(s)	Imp MSE	Rel Bias($\hat{\beta}_1$)	CR($\hat{\beta}_1$)	SE($\hat{\beta}_1$)	SD($\hat{\beta}_1$)
SoftImpute	SI	**7.3**	**0.020**	-0.091	0.78	0.119	0.162
GAIN	SI	39.0	0.868	0.625	0.18	0.146	0.542
Linear RR	SI	3134.7	0.066	0.148	1.00	0.178	0.101
MICE	MI	37.6	**0.023**	**-0.006**	**0.93**	0.116	0.121
Sinkhorn	MI	31.2	0.075	**0.021**	**0.96**	0.186	0.163
MI-GAN$_1$	MI	**3.7**	0.066	**0.027**	0.91	0.157	0.157
MI-GAN$_2$	MI	**8.0**	**0.056**	**0.062**	**0.94**	0.151	0.145
Complete data	-	-	-	**-0.003**	**0.93**	0.109	0.114
Complete case	-	-	-	0.248	0.88	0.340	0.330
ColMean Imp	SI	-	0.141	0.349	0.72	0.221	0.172

TABLE I: Blockwise missing data with $n = 200$ and $p = 251$ under MAR. Approximately **40%** features and **90%** cases contain missing values. Detailed simulation setup information is in Appendix A. Good performance is highlighted in bold.

Models	Style	Time(s)	Imp MSE	Rel Bias($\hat{\beta}_1$)	CR($\hat{\beta}_1$)	SE($\hat{\beta}_1$)	SD($\hat{\beta}_1$)
SoftImpute	SI	12.9	**0.028**	-0.246	0.57	0.137	0.179
GAIN	SI	48.5	0.790	0.697	0.25	0.109	0.727
MICE	MI	32.7	**0.026**	**-0.032**	0.90	0.118	0.141
Sinkhorn	MI	99.8	0.100	-0.193	0.88	0.278	0.326
MI-GAN$_1$	MI	**3.7**	0.076	**-0.004**	0.89	0.188	0.227
MI-GAN$_2$	MI	**8.4**	**0.048**	**0.025**	**0.96**	0.147	0.146
Complete data	-	-	-	**-0.007**	**0.94**	0.111	0.114
Complete case	-	-	-	0.244	0.88	0.376	0.394
ColMean Imp	SI	-	0.135	0.050	0.95	0.357	0.320

TABLE II: Blockwise missing data with $n = 200$ and $p = 501$ under MAR. Approximately **40%** features and **91%** cases contain missing values. Detailed simulation setup information is in Appendix A. Good performance is highlighted in bold.

Models	Style	Time(s)	Imp MSE	Rel Bias($\hat{\beta}_1$)	CR($\hat{\beta}_1$)	SE($\hat{\beta}_1$)	SD($\hat{\beta}_1$)
SoftImpute	SI	28.7	**0.052**	-0.264	0.62	0.166	0.201
GAIN	SI	114.7	0.762	0.906	0.28	0.094	0.720
Sinkhorn	MI	147.9	0.111	-0.070	0.98	0.342	0.290
MI-GAN$_1$	MI	**4.7**	0.099	**-0.035**	**0.95**	0.239	0.233
MI-GAN$_2$	MI	**12.6**	**0.060**	**0.032**	**0.95**	0.162	0.162
Complete data	-	-	-	**-0.010**	**0.96**	0.111	0.114
Complete case	-	-	-	0.311	0.89	0.442	0.462
ColMean Imp	SI	-	0.114	-0.005	1.00	0.351	0.282

TABLE III: Blockwise missing data with $n = 200$ and $p = 1501$ under MAR. Approximately **40%** features and **92%** cases contain missing values. Detailed simulation setup information is in Appendix A. Good performance is highlighted in bold.

For MICE, we use the `iterativeImputer` method in the `scikit-learn` library with default hyperparameters [17]. For MI-GAN$_1$, we use default values of $\lambda_1 = 10$, $\lambda_2 = 0.1$, $m = 256$, $n_{\text{critic}} = 5$, $\alpha = 0.001$, $\beta_1 = 0.5$, and $\beta_2 = 0.9$. For MI-GAN$_2$, we use default values of $N = 3$, $T = 1$, $\lambda_1 = 10$, $\lambda_2 = 0.1$, $m = 256$, $n_{\text{critic}} = 5$, $\alpha = 0.001$, $\beta_1 = 0.5$, and $\beta_2 = 0.9$. Both of MI-GAN$_1$ and MI-GAN$_2$ use shallow multilayer perceptrons (MLP) for generators and discriminators. Specifically, G_k use a four-layer ($p \times p \times p \times p$) MLP with `tanh` activation function and D_k use a four-layer ($p \times p \times p \times 1$) MLP with `ReLU` activation function. MI methods impute missing values for 10 times except GAIN and Linear RR. We notice that the GAIN implementation from its original authors conducts only SI and that Linear RR is computationally very expensive.

A. Synthetic data experiments

In the synthetic data analysis, we generate multiple high-dimensional blockwise missing datasets under MAR mechanism and conduct imputations. Experiment details are included in Appendix A. For each imputation method, we report six performance metrics: imputation mean squared error (denoted by **Imp MSE**), the computing time in seconds per imputation (denoted by **Time(s)**), relative bias of $\hat{\beta}_1$ (denoted by **Rel Bias**($\hat{\beta}_1$)), standard error of $\hat{\beta}_1$ (denoted by **SE**($\hat{\beta}_1$)), coverage rate of the 95% confidence interval for $\hat{\beta}_1$ (denoted by **CR**($\hat{\beta}_1$)), and standard deviation of $\hat{\beta}_1$ across 100 MC datasets (denoted by **SD**($\hat{\beta}_1$)). Here, $\hat{\beta}_1$ is one regression coefficient estimate obtained by fitting a linear regression on the imputed datasets. The first two metrics, Imp MSE and Time(s), are used to measure the imputation accuracy and computational cost. Another three metrics, Rel Bias($\hat{\beta}_1$), SE($\hat{\beta}_1$) and CR($\hat{\beta}_1$), are used to assess statistical inference performance. Of note, CR($\hat{\beta}_1$) that is well below the nominal level of 95% would lead to inflated false positives, an important factor contributing to lack of reproducibility in research. Plus, a well-behaved SE($\hat{\beta}_1$) should be close to SD($\hat{\beta}_1$) and a lower SE/SD denotes a less loss of information.

We summarize in Table I the results over 100 Monte Carlo (MC) datasets on a four-pattern missing data with $n = 200$ and $p = 251$. MICE, Sinkhorn, and MI-GANs show small relative bias of $\hat{\beta}_1$. In addition, MICE, Sinkhorn, and MI-GAN$_2$ yield nearly nominal level of coverage rate for $\hat{\beta}_1$. Notably, only four MI methods, MICE, Sinkhorn and MI-GANs, show well-behaved standard errors. Among them, MICE presents best performance in terms of information recovery due to smallest

SE. Although SoftImpute presents smallest imputation error, it yields poor statistical inference evidenced by large relative bias and well below coverage rate. Similarly, GAIN and Linear RR lead to poor statistical inference. In terms of computational cost, MI-GANs are the most efficient and Linear RR is the most computationally expensive, preventing it to be applicable to higher dimensional settings.

Table II summarizes imputation results on a four-pattern missing data with $n = 200$ and $p = 501$. Since Linear RR costs too much run-time, it is not presented in this table. As we increase the feature size to 501, Sinkhorn's performance degenerates significantly and MICE's performance also deteriorates in terms of CR($\hat{\beta}_1$). In this setting, MICE and MI-GANs show small relative bias of $\hat{\beta}_1$, and only MI-GAN$_2$ yields nearly nominal level of CR($\hat{\beta}_1$). We observe that MI-GAN$_2$ yields much smaller imputation MSE than MI-GAN$_1$, benefiting from its iterative training. Table III summarizes imputation results on a four-pattern missing data as we further increase the feature size to 1501. MICE is not presented due to running out of RAM. At the same time, MI-GANs (especially MI-GAN$_2$) yield satisfactory results in an efficient manner.

B. ADNI data experiments

In the real data analysis, we further evaluate the performance of MI-GANs on a large-scale Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset, which includes both imaging and gene expression data. The original dataset contains 649 cases; each case contains more than 19000 features and a continuous response variable — the VBM right hippocampal volume. After standardizing each feature, 1000 features that we are interested in are selected as the experiment dataset; among them, three features, which have the maximal correlation with the response variable, are selected as predictors for the subsequent linear regression.

Here we summarize the results over 100 repeats. Experiments details is included in Appendix B. Notice that we have no access to the true regression coefficient $\boldsymbol{\beta}$ in real data analysis. Hence we instead report four metrics: **Imp MSE**, **Time(s)**, $\hat{\beta}_1$ and **SE(**$\hat{\beta}_1$**)** for each method.

Table IV presents the results for this dataset. MICE and Linear RR are not presented due to running out of RAM. Although SoftImpute yields the smallest imputation MSE, its $\hat{\beta}_1$ estimate is far away from the golden standard (which is the $\hat{\beta}_1$ estimate from complete data analysis). Besides GAIN, MI-GAN$_1$ and MI-GAN$_2$ yields the $\hat{\beta}_1$ estimate closest to the golden standard, which shows MI-GANs can lead to good statistical inference. However, GAIN yields much higher and unacceptable imputation MSE than the naive approach, ColMean Imp, indicating that GAIN is not regarded as a good imputation method in this high-dimensional setting. In addition, MI-GANs is the most computationally efficient, compared to all other state-of-the-art methods. Taking all metrics into consideration, MI-GANs are overall the most powerful imputation method on this setting.

Models	Style	Time(s)	Imp MSE	$\hat{\beta}_1$ Value	SE($\hat{\beta}_1$)
SoftImpute	SI	146.8	**0.057**	0.027	0.012
GAIN	SI	84.4	1.264	**0.018**	0.009
Sinkhorn	MI	219.2	0.076	0.026	0.012
MI-GAN$_1$	MI	**6.3**	**0.063**	0.025	0.011
MI-GAN$_2$	MI	**17.3**	**0.075**	**0.022**	0.012
Complete data	-	-	-	**0.016**	0.008
Complete case	-	-	-	0.025	0.017
ColMean Imp	SI	-	0.177	**0.021**	0.013

TABLE IV: Real data experiment with $n = 649$ and $p = 1001$ under MAR. Approximately **40%** features and **75%** cases contain missing values. Linear RR and MICE are not included as they run out of RAM. Detailed experiment information is in Appendix B. Good performance is highlighted in bold.

IV. Discussion

In this work, we propose a novel GAN-based multiple imputation method, MI-GANs, which can handle high-dimensional blockwise missing data with theoretical support under MAR/MCAR mechanism. Our experiments demonstrate that MI-GANs compete with current state-of-the-art imputation methods and outperform them in the sense of statistical inference and computational speed. One limitation of MI-GANs is that GAN's training is challenging and the generators may not converge when the training sample size is too small. One potential research interest is applying graph neural networks in the generators and the discriminators to reduce the number of parameters when the knowledge graph is available. This may help the generators converge to the optimal point and learn the true conditional distribution.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[2] S van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, pages 1–68, 2010.

[3] Yuxin Chen, Jianqing Fan, Cong Ma, and Yuling Yan. Inference and uncertainty quantification for noisy matrix completion. *Proceedings of the National Academy of Sciences*, 116(46):22931–22937, 2019.

[4] Yi Deng, Changgee Chang, Moges Seyoum Ido, and Qi Long. Multiple imputation for general missing data patterns in the presence of high-dimensional data. *Scientific reports*, 6(1):1–10, 2016.

[5] Pedro J García-Laencina, José-Luis Sancho-Gómez, and Aníbal R Figueiras-Vidal. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19(2):263–282, 2010.

[6] Lovedeep Gondara and Ke Wang. Mida: Multiple imputation using denoising autoencoders. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 260–272. Springer, 2018.

[7] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[8] Trevor Hastie, Rahul Mazumder, Jason D Lee, and Reza Zadeh. Matrix completion and low-rank svd via fast alternating least squares. *The Journal of Machine Learning Research*, 16(1):3367–3402, 2015.

[9] Oleg Ivanov, Michael Figurnov, and Dmitry Vetrov. Variational autoencoder with arbitrary conditioning. *arXiv preprint arXiv:1806.02382*, 2018.

[10] Dongwook Lee, Junyoung Kim, Won-Jin Moon, and Jong Chul Ye. Collagan: Collaborative gan for missing image data imputation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2487–2496, 2019.

[11] Steven Cheng-Xian Li, Bo Jiang, and Benjamin Marlin. Misgan: Learning from incomplete data with generative adversarial networks. *arXiv preprint arXiv:1902.09599*, 2019.

[12] Faming Liang, Bochao Jia, Jingnan Xue, Qizhai Li, and Ye Luo. An imputation–regularized optimization algorithm for high dimensional missing data problems and beyond. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(5):899–926, 2018.

[13] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

[14] Pierre-Alexandre Mattei and Jes Frellsen. Miwae: Deep generative modelling and imputation of incomplete data sets. In *International Conference on Machine Learning*, pages 4413–4423. PMLR, 2019.

[15] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11:2287–2322, 2010.

[16] Boris Muzellec, Julie Josse, Claire Boyer, and Marco Cuturi. Missing data imputation using optimal transport. *arXiv preprint arXiv:2002.03860*, 2020.

[17] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[18] Joseph L Schafer. *Analysis of incomplete multivariate data*. CRC press, 1997.

[19] Nathan Srebro, Jason Rennie, and Tommi Jaakkola. Maximum-margin matrix factorization. *Advances in neural information processing systems*, 17:1329–1336, 2004.

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

[21] Stef Van Buuren. Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical methods in medical research*, 16(3):219–242, 2007.

[22] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[23] Jinsung Yoon, James Jordon, and Mihaela Van Der Schaar. Gain: Missing data imputation using generative adversarial nets. *arXiv preprint arXiv:1806.02920*, 2018.

[24] Hongyang Zhang and David P Woodruff. Medical missing data imputation by stackelberg gan. *Carnegie Mellon University*, 2018.

[25] Yize Zhao and Qi Long. Multiple imputation in the presence of high-dimensional data. *Statistical Methods in Medical Research*, 25(5):2021–2035, 2016.

[26] Yuxuan Zhao and Madeleine Udell. Missing value imputation for mixed data via gaussian copula. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 636–646, 2020.

Appendix

A. Synthetic data experiments

Each MC dataset contains $n = 200$ samples and each sample contains p features including $(p-1)$ predictors or auxiliary variables $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{p-1})$ and a response variable y. Here we consider settings where $p = 251$, $p = 501$ and $p = 1501$. $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{p-1})$ is generated by reordering variables $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_{p-1})$. $\mathbf{A}$ is a first order autoregressive model with autocorrelation $\rho = 0.9$, white noise $\epsilon \sim \mathcal{N}(0, 0.1^2)$ and $\mathbf{a}_1 \sim \mathcal{N}(0,1)$. Given a variable vector $\mathbf{A}$, $\mathbf{X}$ is obtained by firstly moving a_{5k+4} ($k \in \mathbb{N}$) to the right, secondly moving a_{5k+5} ($k \in \mathbb{N}$) to the right (for example, if $p = 11$, $(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5, \mathbf{a}_6, \mathbf{a}_7, \mathbf{a}_7, \mathbf{a}_9, \mathbf{a}_{10})$ becomes $\mathbf{X} = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_6, \mathbf{a}_7, \mathbf{a}_8, \mathbf{a}_4, \mathbf{a}_9, \mathbf{a}_5, \mathbf{a}_{10})$). Given $\mathbf{X}$, the response $\mathbf{y}$ is generated by

$$\mathbf{y} = \beta_1 \cdot \mathbf{x}_{q[1]} + \beta_2 \cdot \mathbf{x}_{q[2]} + \beta_3 \cdot \mathbf{x}_{q[3]} + \mathcal{N}(0, \sigma_1^2) \tag{2}$$

where $\beta_i = 1$ for $i \in \{1,2,3\}$ and q is the predictor set. For $p = 251, 501, 1501$, the corresponding predictor sets are $\{210, 220, 230\}$, $\{380, 400, 420\}$, and $\{1100, 1200, 1300\}$. Missing values are separately generated in $\{\mathbf{x}_{\frac{3}{5}(p-1)+1}, \ldots, \mathbf{x}_{\frac{4}{5}(p-1)}\}$ and $\{\mathbf{x}_{\frac{4}{5}(p-1)+1}, \ldots, \mathbf{x}_{p-1}\}$ from MAR. Specifically, suppose their missing indicators are $\mathbf{R}_1$ and $\mathbf{R}_2$, then

$$\text{logit}(\mathbb{P}(\mathbf{R}_1 = 1|\mathbf{X}, \mathbf{y})) = 1 - 2 \cdot \frac{5}{3(p-1)} \sum_{j=1}^{3(p-1)/5} \mathbf{x}_j + 3 \cdot \mathbf{y} \tag{3}$$

$$\text{logit}(\mathbb{P}(\mathbf{R}_2 = 1|\mathbf{X}, \mathbf{y})) = 2 \cdot \frac{5}{3(p-1)} \sum_{j=1}^{3(p-1)/5} \mathbf{x}_j - 2 \cdot \mathbf{y} \tag{4}$$

Here $\mathbf{R}_i = 1$ indicates the corresponding group of variables is missing.

B. ADNI data experiments

1) Data Availability: The de-identified ADNI dataset is publicly available at http://adni.loni.usc.edu/.

2) Experiment details: The original large-scale dataset contains 649 samples and each sample contains 19823 features including a response variable $(\mathbf{y})$, the VBM right hippocampal volume. We prepocess features except response $\mathbf{y}$ by removing their means. Then we rearrange these features in the decreasing order of correlation with $\mathbf{y}$ and only select the first 1000 features, namely $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_{1000})$, to analyze. For each repeat of experiment, we randomly generate missing values in two groups: $\{\mathbf{x}_1, \ldots, \mathbf{x}_{200}\}$ and $\{\mathbf{x}_{201}, \ldots, \mathbf{x}_{400}\}$. Their missing indicators $\mathbf{R}_1, \mathbf{R}_2$ are generated from MAR:

$$\text{logit}(\mathbb{P}(\mathbf{R}_1 = 1)) = -1 - \frac{3}{100} \sum_{j=401}^{500} \mathbf{x}_j + 3\mathbf{y}$$

$$\text{logit}(\mathbb{P}(\mathbf{R}_2 = 1)) = -1 - \frac{3}{100} \sum_{j=601}^{700} \mathbf{x}_j + 2\mathbf{y}$$

Here $\mathbf{R}_i = 1$ indicates the corresponding group of variables is missing. After imputing the missing values, we fit a linear regression $\mathbb{E}[\mathbf{y}|\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3] = \beta_0 + \beta_1\mathbf{x}_1 + \beta_2\mathbf{x}_2 + \beta_3\mathbf{x}_3$ and analyze the coefficient β_1.