Data Synthesis Reinvented: Preserving Missing Patterns for Enhanced Analysis

Xinyue Wang*, Hafiz Asif†, Shashank Gupta‡, Jaideep Vaidya‡
*Renmin University, Beijing, China
†Hofstra University, Long Island, NY, USA
‡Rutgers University, Newark, NJ, USA
Email: xinyue.wang@ruc.edu.cn, hafiz.asif@hofstra.edu, shashank.and.gupta@gmail.com, jsvaidya@rutgers.edu

***Abstract*—Synthetic data is being widely used as a replacement or enhancement for real data in fields as diverse as healthcare, telecommunications, and finance. Unlike real data, which represents actual people and objects, synthetic data is generated from an estimated distribution that retains key statistical properties of the real data. This makes synthetic data attractive for sharing while addressing privacy, confidentiality, and autonomy concerns.**

Real data often contains missing values that hold important information about individual, system, or organizational behavior. Standard synthetic data generation methods eliminate missing values as part of their pre-processing steps and thus completely ignore this valuable source of information. Instead, we propose methods to generate synthetic data that preserve both the observable and missing data distributions; consequently, retaining the valuable information encoded in the missing patterns of the real data. Our approach handles various missing data scenarios and can easily integrate with existing data generation methods. Extensive empirical evaluations on diverse datasets demonstrate the effectiveness of our approach as well as the value of preserving missing data distribution in synthetic data.

***Index Terms*—Synthetic Data Generation, GAN, Missing Data**

I. Introduction and Related Work

Modern machine learning and artificial intelligence cannot be done without access to data. Indeed, many of the recent successes in this field, such as generative AI and large language models have become possible only due to their training on humongous amounts of data. However, in many domains and situations where real data contains sensitive information (e.g., health or user behaviorrelated data), or when raw data cannot be shared due to legislative requirements (e.g., GDPR, HIPAA, and CCPA), synthetically generated data is used as a privacy-preserving replacement for real data [1], [2]. Many methods have been developed to generate synthetic data (such as [3], [4], [5]), however all of them assume that the underlying real data is complete, i.e., without any missing values. This is often not the case, with missingness as high as 99% in e-commerce and social media datasets.

More importantly, the missing data can have a lot of value, sometimes even more than the observable data. An anecdotal example comes from World War II [6], when the airforce wanted to reinforce bombers with more armor to prevent them from being shot down. However, additional armor would increase weight and result in less bomb carrying capacity. Therefore, the airforce tasked the Statistical Research Group (SRG) to decide the amount of armor to put in different locations based on the statistical data regarding aircraft damage, indicated by the distribution of the bullet holes on the returning planes, i.e., the observable data. Interestingly, Abraham Wald, the mathematician in charge of solving this problem, made the keen observation that the *missing data* provided the crucial information to solve this problem. Specifically, the data about the possible bullet holes on the missing planes, the ones that got shot down and did not return, was more important because the observable data showed the survivable parts of the plane. Based on Wald's hypothesis, more armor was placed on the engines, contrary to the observable data which showed that engines were less damaged. This actually resulted in a significant increase in the number of airplanes returning to base, demonstrating the value of the missing data.

Similarly, in many practical settings, missing data is crucial for effective data analysis. Typically, data is not missing completely at random (MCAR). Instead, the missingness is often due to underlying data, system, or behavior-dependent mechanisms that capture complex situational or environmental interactions [7], [8].For instance, the missing answers about smoking on the medical chart may not be due to the question not being asked, but rather because pregnant women with depression are less likely to report smoking than non-depressed pregnant women [9]. The missing gender and COVID-19 test related information in symptoms-tracking surveys is strongly linked with users' privacy preferences that are unobservable without the missing data [10].

Since the standard synthetic data generation methods operate on data without any missing values, if the real data has missing values, either all of the records with missing values are removed (known as complete-case analysis) or the missing data is imputed. Complete-case analysis [11], [2] not only throws away the missing data, but also ignores the observable data in incomplete samples. Hence, it is clearly unfit for all the real-life situations where missing values are data dependent – i.e., Missingness at Random (MAR) or Missingness Not at Random (MNAR) settings [12]. While the alternative approach of imputing the missing data before generation [13], [14] better utilizes data correlations and does not throw away a lot of data, it still hides all the missing information from the downstream data analyst receiving the synthetic data.

This work was supported by the NIH under award R35GM134927 and by the NSF under award CNS-2333225. The content is solely the responsibility of the authors and does not necessarily represent the official views of the agencies funding the research.

Effectively, both the existing approaches (complete-case analysis and impute-then-generate) erase all patterns and structures related to missingness, losing important information, which can be leveraged in many settings (e.g., in healthcare and bioinformatics) to solve a number of problems[7]. Indeed, data analysts employ various methods to leverage missing data to enhance the performance of their specific downstream tasks, methods that are often unknown to the data generator. Analysts can achieve better imputation by utilizing their domain expertise and additional information [15], [16], which the data generator might lack. They can also use the distribution of missing values as an extra signal to refine their models, such as in recommendation systems[17], [18]. Moreover, many learning algorithms explicitly account for missing data and have outperformed those that either ignore or impute missing values [19], [20], [21], [22].

As discussed above, since missingness is an integral part of the data, this article aims to answer the question: "*Given real data with missing values, how can we generate synthetic data that accurately reflects both the distribution of the observable data and the distribution of the missing data?*" The key challenge here is to either explicitly or implicitly model, learn, and sample from the joint distribution of the observable and missing data, regardless of the underlying heterogeneity, complexity, or interaction of the missingness mechanisms.

Although some recent works attempt to address the aforementioned challenge, they do so for a very restricted setting [23] or are unable to faithfully reproduce missing data distribution in the synthetic data [24]. Thus, in this work, we make the following key contributions:

1) We initiate the systematic study of approaches to generate synthetic data that maintain both observable and missing data distributions.
2) We propose several alternative methods to generate high quality synthetic data and theoretically analyze each methods providing recommendations for when each is likely to perform well.
3) Our methodology can be easily integrated into existing data generation pipeline as a pre-processing step, and it is independent of any specific data generation method.
4) We demonstrate the effectiveness of our methods in preserving the two distributions by providing an extensive empirical evaluation comparing against several baselines in terms of both statistical metrics and downstream tasks over a range of fabricated and real-world datasets.

A preliminary version of this paper presenting two alternative approaches was published in [25]. This article significantly extends that work developing an additional hybrid approach which outperforms both approaches in [25]. Additionally, the article includes a thorough theoretical analysis as well as an enhanced comparative analysis with a more robust empirical evaluation with additional baselines.

II. Related Work

Synthetic data generation is used in many fields for a wide variety of data. There are many techniques to generate synthetic data, for example, sampling from random forest [2], via dynamic time wrapping Barycentric averaging [26], Hidden Markov Models [27], probabilistic database model [28], Markov random fields [29], and generative adversarial networks (GANs) [30], [11], [3]. *GANs* have recently achieved a new state-of-the-art performance for synthetic tabular data and imputation of missing data. TGAN [31], for instance, is a synthetic data generator for tabular data with mixed variable types. They use Long Short Term Memory (LSTM) to generate data column by column. In [32], several extensions to TGAN are proposed to support database constraints in synthetic data generation. CTGAN [33] is a newer GAN-based method to generate tabular data. It can handle imbalanced discrete columns by the conditional generator and training-by-sampling technique and generate high-quality data. However, *both models cannot handle missing data*; so they impute the data first (e.g., by using GAIN [34], a novel GAN-based imputation method).

Synthetic data generation while preserving missing data distribution has not received much attention in the literature. Most work, in contrast, focuses on getting rid of the missing data either by imputation [35], [36] or deleting all the records with missing values (i.e., complete-case analysis). Only very recently a GAN-based model, named MisGAN[23], was proposed to generate synthetic data with missing values for image data. In this work, two GANs are used, one to learn the missing pattern (referred to as masks) distribution and the other to learn masked data (where the missing values are replaced by a constant) distribution. They empirically showed that synthetic data with missing values (by MisGAN) gives better statistical estimations. MisGAN, however, has some serious limitations: it only works for MCAR setting, i.e., missingness is independent of the data, an assumption often violated by real world datasets; applying MisGAN to tabular data is not a trivial task. So, we propose a suite of methods to deal with various missing mechanisms, such as MCAR, MAR, and MNAR. Additionally, we also formulate a measure to assess the quality of synthetic data with missing values. Another work uses Bayesian Networks to generate synthetic data with missing values [24]. However, their empirical results show a very large divergence in missing pattern distribution and other statistics. For instance, for "cholesterol" feature with the original missing rate of 88.59%, the corresponding synthetic data had the missing rate of 0.08%. Further, it fails to scale to high dimensional and large scale data.

III. Background and Notation

A. Data, samples, and distributions

Let $\mathcal{D} = \prod_{t=1}^{d} F_t$ denote the data universe, consisting of the set of all possible samples (i.e., records) with and without the missing values of all features, where F_t is the set of possible values of the t-th feature; each feature has a special value `NA` to denote that the feature's value is missing (i.e., $\texttt{NA} \in F_t$ for each t). We use $\mathcal{D}_o$ to denote data universe with *fully observable* records, i.e., without any missing values. The dimension d is arbitrarily fixed throughout this work. $\overline{\boldsymbol{x}} = \{x_1, x_2, \ldots, x_n\}$ denotes the dataset (a multi-set) of n samples, where each sample x_i is a d-dimensional vector.

Thus, for any sample x (we omit the subscript when the distinction is not needed), if the value of its j-th feature is missing, we have $x_j = \texttt{NA}$. $|\overline{\boldsymbol{x}}|$ denotes the size of the dataset $\overline{\boldsymbol{x}}$ (e.g., here $|\overline{\boldsymbol{x}}| = n$).

We use bold lower case letter with a bar to denote a dataset ($\overline{\boldsymbol{x}}$) and upper case letter to denote a random variable (X and X^o). $\mathbb{P}(X)$ denotes the distribution that X follows, we also use $\mathbb{P}(\cdot)$ to denote $\Pr(\cdot)$ as described below.

The samples in $\overline{\boldsymbol{x}}$ have been sampled from data distribution independently and identically (*i.i.d.*), and we use X to denote the random variable for a sample. Now $\mathbb{P}(X)$ gives the data distribution, which is a probability density function or probability mass function depending upon the underlying space. Since our aim is to develop methods for generating synthetic data that learn the data distribution from a given dataset, we use $\mathbb{P}_{\overline{\boldsymbol{x}}}(X)$ to denote the learned data distribution using the dataset $\overline{\boldsymbol{x}}$ (i.e., $\mathbb{P}_{\overline{\boldsymbol{x}}}(X)$ is a dataset-specific approximation of $\mathbb{P}(X)$). Note that for any measurable event $E \subseteq \mathcal{D}$, $\mathbb{P}(E) = \Pr[x \in E]$ gives the probability of the event. Since $\overline{\boldsymbol{x}}$ may or may not contain missing values, for modeling and definitions, we also consider a random variable X^o, which is the counterpart of X and corresponds to sampling data *without* any missing values. Similar to the above, $\mathbb{P}(X^o)$ gives the distribution for the counterpart of our sampled dataset, which does not have any missing values (this is useful in describing missing mechanisms which model generation of missing values in the data, discussed below). Indeed X^o and X are related via the missing mechanism, as described next.

B. Missingness and Missing Mechanisms

Although the missing values appear at feature level, the appropriate level of generalization is at the sample/record level, which we refer to as *missing pattern* of a sample. This will be a central notion throughout this work for it is a crucial characteristic of the real-world datasets (discussed in Section V). A missing pattern (*mp*) describes which features of a sample are missing and can be modeled as binary vector: thus, we use $\mathcal{MP} = \{0,1\}^d$ to denote the set of all missing patterns for any d-dimensional dataset. Hence, for any sample $x \in \mathcal{D}$, we say $m \in \mathcal{MP} = \{0,1\}^d$ is x's missing pattern if $m_i = 1$ when $x_i = \texttt{NA}$ and $m_i = 0$ when $x_i \neq \texttt{NA}$ (for every $i = 1, 2, \ldots, d$). Now for a missing pattern m, we conveniently use $\overline{\boldsymbol{x}}_m$ to denote the set of all the samples in $\overline{\boldsymbol{x}}$ that have the missing pattern m (i.e., $\overline{\boldsymbol{x}}_m = \{x \text{ in } \overline{\boldsymbol{x}} \mid \texttt{miss-patt}(x) = m\}$, where `miss-patt` gives the missing pattern of x).

Using missing patterns, we define missing pattern (*mp*)-*distribution* for the data sampled according to $\mathbb{P}(X)$, which we denote as $\mathbb{P}(M)$, where M is the random variable for the missing patters for the sampled data. Since $\mathcal{MP}$ is finite (or at max countable), $\mathbb{P}(M)$ is the probability mass function. For a given $\overline{\boldsymbol{x}}$, we use $\mathbb{P}_{\overline{\boldsymbol{x}}}(M)$ to denote the empirical *mp*-distribution as per $\overline{\boldsymbol{x}}$: so, for any $m \in \mathcal{MP}$, $\mathbb{P}_{\overline{\boldsymbol{x}}}(M = m) = |\overline{\boldsymbol{x}}_m|/|\overline{\boldsymbol{x}}|$. We write $\mathbb{P}_{\overline{\boldsymbol{x}}}(m) = \mathbb{P}_{\overline{\boldsymbol{x}}}(M = m)$ for brevity. Note that since there are an exponential (in d) number of missing patterns, in typical settings, $\mathbb{P}_{\overline{\boldsymbol{x}}}(m) = 0$ for many patterns m, since there may be no record in the dataset with that missing pattern.

Let us now introduce the notion of missing mechanism, a standard way to model missingness in the data: missing mechanism is an abstraction of the underlying process responsible for creating missing values in the data [37]. It is conceptualized as an algorithm that takes in fully observable data and then creates missing value by omitting some value or replacing them with NA's. These mechanisms are divided into three categories, which are based on how the missing vales are related to the data [37]. In order to understand missing mechanisms, it is helpful to describe data through a pair (X, X_{hid}) of two random variables, where X, termed as *observable data*, consists of collected features with possible missing values (as described in Section III-A), and X_{hid} is the hidden part, which consists of the uncollected features that are relevant to missing values as well as the feature values that are missing; for instance, consider data on individuals about an infectious disease, which typically does not have the level of individuals' exposure. Using random variables (X, X_{hid}) and M, we define the three mechanisms as:

- A missing mechanism is called MCAR if the missing values created by it are *missing completely at random (MCAR)*, namely, they are independent of the data, i.e., $\mathbb{P}(M \mid (X, X_{hid})) = \mathbb{P}(M)$.
- An MAR mechanism creates *missing values at random (MAR)*, namely, missingness depends on the part of the data that is observable (not missing), i.e., $\mathbb{P}(M \mid (X, X_{hid})) = \mathbb{P}(M \mid X)$.
- An MNAR mechanism creates *missingness not at random (MNAR)*, namely, missing values creation depends on the observable data as well as on the hidden data, and hence, it is neither MCAR nor MAR, i.e., $\mathbb{P}(M \mid (X, X_{hid})) \neq \mathbb{P}(M \mid X)$ and $\mathbb{P}(M \mid (X, X_{hid})) \neq \mathbb{P}(M)$.

C. Synthetic Data Generation Methods (DGMs)

For synthetic data generation, we consider machine learning based generative methods, such as generative adversarial networks (GANs) [30], variational autoencoders (VAEs) [38], Bayesian networks (BNs) [39]. We refer to them as *data generation methods (DGMs)*. Typically, a DGM, takes as input a dataset that it uses to learn the underlying data distribution as a generator. The learned generator is subsequently used to generate synthetic data samples. Since only samples generated from the learned distribution are released, these methods are believed to provide statistical disclosure control, and hence, are considered privacy protecting in various practical scenarios [27], [40]. When stronger privacy guarantees are needed, one can use the rigorous framework of differential privacy to provably protect privacy during the learning process [41].

DGMs can be broadly classified into two groups: traditional statistical methods and deep generative models. Statistical methods necessitate an a priori model, viewing real data as an instance of random variables following an inherent probabilistic distribution. Synthetic data is then sampled from the built model. For instance, Bayesian Networks are used for learning the joint probability distributions [39]. In contrast, deep learning-based approaches aim to model the underlying data structure in a data-driven manner without needing predefined

mathematical equations or simulations. This category includes models like VAEs [38], which compress data into a lower-dimensional latent space before reconstructing it, and are utilized across various domains such as imaging and tabular data. Another prominent DGM method is based on GANs [30], which consist of two adversarially related networks that are trained together, where they try to beat each other in a game, summarized as the minimization of a common loss function.

In this paper, we employ CTGAN [33], vanilla VAE, and Bayesian Network to demonstrate our synthetic data generation approach. These models are adept at handling both continuous and discrete data – this is important as we aim to build a method to generate synthetic data that can handle mixed data types.

***Wasserstein distance**:* To assess the quality of the synthetic data, we will make use of Wasserstein distance [42]. In machine learning, it is often used to measure how close two datasets (i.e., sets of samples) (or distributions) are [42]. Compared to Kullback-Leibler or Jensen-Shannon divergences, it is sensitive to small discrepancies and more suitable in scenarios where distributions have differing supports or are sparse, making it an effective tool for evaluating synthetic data quality. For two datasets, $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{y}}$, each of size n, (quadratic) Wasserstein distance is defined as follows:

$$W^2(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) = \frac{1}{n} \min_{\pi \in \Pi_n} \sum_{i=1}^{n} \left|\left| x_{\pi(i)} - y_i \right|\right|^2, \tag{1}$$

where Π is the set of all permutations ($\pi : [n] \to [n]$, for $[n] = \{1, 2, \ldots, n\}$). Calculating Wasserstein distance over high dimensional data is computationally very resource intensive and time consuming; thus, we will employ its sliced version, which computes the distance as the average over several random linear projection [43]. Since Wasserstein and many other distance notion do not work when the data samples consist of missing values, especially, when used to measure quality of the generated data, one of the contributions of this work is to propose methodologies to tackle this problem gracefully, as described next.

IV. Assessing Synthetic Data Quality

We aim to develop approaches for synthetic data generation that maintain the fidelity not only in terms of the observable data but also missing data. The generation method will take as input a dataset $\overline{\boldsymbol{x}}$, call it *real data*, possibly with missing values, and generate *synthetic dataset* $\overline{\boldsymbol{y}}$ such that $\overline{\boldsymbol{y}}$ is similar to $\overline{\boldsymbol{x}}$, in their observable and missing data distributions. In this section, we propose a way to measure and assess these two important but contrasting* properties of the data.

We achieve the aforementioned goal by considering (empirical) missing pattern (*mp*)-distributions corresponding to the two datasets, and then define a notion of (α, β)-closeness (Definition 1) where α bounds closeness in terms of *mp*-distributions and β bounds the closeness in terms of the observable data while respecting the missing data distribution.

*The contrast is due to the fact that one property, i.e., observable data distribution, is about what we can see and the other property, i.e., missing data distribution, is about what we cannot see.

Note that the intuition and understanding gleaned from the explication of the consolidated measure directly guides the development of appropriate approaches for high fidelity synthetic data generation (in Section V).

A. Measuring similarity in terms of mp-distribution

For a given dataset $\overline{\boldsymbol{x}}$, $\mathbb{P}_{\overline{\boldsymbol{x}}}(M)$ denotes the empirical *mp*-distribution per $\overline{\boldsymbol{x}}$. Recall that $\mathbb{P}(X)$ and $\mathbb{P}(M)$ give the actual probability distributions of an i.i.d. sample $x \in \mathcal{D}$ and its missing pattern $m \in \mathcal{MP} = \{0, 1\}^d$ respectively with their corresponding random variables X and M (where $m_j = 1$ if j-th feature in x is missing, see Section III-A and III-B for details). Now for any given datasets $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{y}}$, the closeness between their corresponding *mp*-distributions, measured via L_1 norm, is as follows:

$$D_{mis}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) = ||\mathbb{P}_{\overline{\boldsymbol{x}}}(M) - \mathbb{P}_{\overline{\boldsymbol{y}}}(M)\,||_1 = \sum_{m \in \mathcal{MP}} \delta_m(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}), \tag{2}$$

where $\delta_m(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) = |\mathbb{P}_{\overline{\boldsymbol{x}}}(m) - \mathbb{P}_{\overline{\boldsymbol{y}}}(m)\,|$ and $\overline{\boldsymbol{x}}_m$ ($\overline{\boldsymbol{y}}_m$) are the subsets of $\overline{\boldsymbol{x}}$ ($\overline{\boldsymbol{y}}$), which consist of all the samples with missing patterns m, and $\mathbb{P}_{\overline{\boldsymbol{x}}}(m) = \mathbb{P}_{\overline{\boldsymbol{x}}}(M = m)$.

B. Defining similarity between $\overline{\boldsymbol{x}}_m$ and $\overline{\boldsymbol{y}}_m$

Then, we describe how to measure the similarity between $\overline{\boldsymbol{x}}_m$ and $\overline{\boldsymbol{y}}_m$, $\mathcal{W}(\overline{\boldsymbol{x}}_m, \overline{\boldsymbol{y}}_m)$, especially, when $|\overline{\boldsymbol{x}}_m| \neq |\overline{\boldsymbol{y}}_m|$. Note that sliced Wasserstein distance is able to handle such cases as long as $|\overline{\boldsymbol{x}}_m|$ and $|\overline{\boldsymbol{y}}_m|$ are sufficiently large [44].

We measure $\mathcal{W}$ as the average of the given similarity measure over equal sized subsets. Below, we describe this for Wasserstein distance W.

To do this, we exploit the fact that W can be naturally extended to measure the distance between two non-empty and equal sized datasets with same missing pattern by projecting all the samples down to only non-missing features, e.g., $(1, \texttt{NA}, 3) \to (1, 3)$. Our formulation not only resolves all the issues we discussed earlier but also gives a useful relationship between S and Wasserstein distance. For instance, $S_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) = W(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$ when both the datasets consist of the same single missing pattern; $S_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$ gives non-uniform weighted version of the Wasserstein distance when $D_{mis}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) = 0$ and $|\overline{\boldsymbol{x}}_m| = |\overline{\boldsymbol{y}}_m|$ for each m.

Hence, for two given datasets, $\overline{\boldsymbol{x}}_m$ of size N and $\overline{\boldsymbol{y}}_m$ of size K, with the same missing pattern m, we define $\mathcal{W}$ as follows:

$$\mathcal{W}^2(\overline{\boldsymbol{x}}_m, \overline{\boldsymbol{y}}_m) = \begin{cases} 0 & N = K = 0 \\ W^2(\overline{\boldsymbol{x}}_m, \overline{\boldsymbol{y}}_m) & N = K > 0 \\ \binom{N}{K}^{-1} \sum\limits_{\overline{\boldsymbol{z}} \in C_{\overline{\boldsymbol{x}}, K}} W^2(\overline{\boldsymbol{z}}, \overline{\boldsymbol{y}}_m) & N > K > 0 \\ \mathcal{W}^2(\overline{\boldsymbol{y}}_m, \overline{\boldsymbol{x}}_m) & K > N > 0 \\ \mathcal{W}^2(\overline{\boldsymbol{x}}_m, \{\mathbf{0}_m\}) & K = 0 \\ \mathcal{W}^2(\overline{\boldsymbol{y}}_m, \overline{\boldsymbol{x}}_m) & N = 0 \end{cases} \tag{3}$$

where $\mathbf{0}_m$ is the zero vector with NA's according to m; and $C_{\overline{\boldsymbol{x}}, K}$ is the set of all datasets of size K that can be created from samples in $\overline{\boldsymbol{x}}_m$, that is, for $\overline{\boldsymbol{x}}_m = \{x_1, \ldots, x_N\}$, $C_{\overline{\boldsymbol{x}}, K} = \{x_i \text{ in } \overline{\boldsymbol{x}}_m \mid i \in I \subseteq [N] \text{ s.t. } |I| = K\}$, and hence, its size is $\binom{N}{K}$.

C. Closeness measure between $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{y}}$

We now discuss our proposed consolidated measure to characterize closeness between $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{y}}$, which in addition to the observable data, takes into account datasets' differences in terms of their *mp*-distributions. Note that the variations in the missing patterns across the two datasets and their unequal sizes make it impossible to apply the existing measure (such as Wasserstein distance from Section III-C) directly over all the samples. For example, two samples with different missing patterns can have different dimensions (in terms of observable features), and thus, cannot be compared as such — e.g., consider comparing $(1, \texttt{NA}, 5)$ to $(\texttt{NA}, 9, \texttt{NA})$.

To solve this conundrum, we propose that: firstly, a similarity scoring function, s, be used to measure the similarity between samples with the same missing pattern m (i.e., $\overline{\boldsymbol{x}}_m$ and $\overline{\boldsymbol{y}}_m$) from the two datasets ($\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{y}}$); and then, $S(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$ is defined as a weighted average of these scores. This approach allows the use of standard metrics, distance functions, and similarity measures for s[†], which can now be applied over the non-missing coordinates as they remain non-missing in all the samples with same missing pattern, e.g., we can compute Euclidean distance between $(7, \texttt{NA}, 3)$ and $(2, \texttt{NA}, 1)$ by ignoring missing (i.e., NA) coordinate. Thus, for a given similarity function s and non-negative weights, γ, we define:

$$S(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) = \sqrt{\sum_{m \in \mathcal{MP}} \gamma_m \cdot s^2(\overline{\boldsymbol{x}}_m, \overline{\boldsymbol{y}}_m)}, \tag{4}$$

where γ_m gives the weights corresponding to the missing pattern m. In this work, we define similarity between $\overline{\boldsymbol{x}}_m$ and $\overline{\boldsymbol{y}}_m$ as an average Wasserstein distance, $\mathcal{W}$ (given by Eq. (3)), i.e., $s(\overline{\boldsymbol{x}}_m, \overline{\boldsymbol{y}}_m) = \mathcal{W}(\overline{\boldsymbol{x}}_m, \overline{\boldsymbol{y}}_m)$. Further, we define weights γ with respect to a reference dataset $\overline{\boldsymbol{z}}$ such that for every $m \in \mathcal{MP}$, we have $\gamma_m = \mathbb{P}_{\overline{\boldsymbol{z}}}(m) + \delta_m(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$; This implies:

$$S_{\overline{\boldsymbol{z}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) = \sqrt{\sum_{m \in \mathcal{MP}} (\mathbb{P}_{\overline{\boldsymbol{z}}}(m) + \delta_m(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})) \cdot \mathcal{W}^2(\overline{\boldsymbol{x}}_m, \overline{\boldsymbol{y}}_m)}. \tag{5}$$

Above we use $\overline{\boldsymbol{z}}$ as the subscript to denote the use of our specific weight function. Note that since $\mathcal{W}$ gives average distance, the smaller the value of $S_{\overline{\boldsymbol{z}}}$, the higher the similarity. We now define (α, β)-closeness as follows:

Definition 1 ((α, β)-closeness): For given $\alpha, \beta \geq 0$ and a (real) dataset $\overline{\boldsymbol{x}}$, we say a (synthetic) dataset $\overline{\boldsymbol{y}}$ is (α, β)-close to $\overline{\boldsymbol{x}}$ if

$$D_{mis}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) \leq \alpha \text{ and } S_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) \leq \beta.$$

Note that the reference dataset (above) is fixed as the given dataset, i.e., $\overline{\boldsymbol{z}} = \overline{\boldsymbol{x}}$. Thus, giving $\gamma_m(\overline{\boldsymbol{x}}_m, \overline{\boldsymbol{y}}_m) = \mathbb{P}_{\overline{\boldsymbol{x}}}(m) + \delta_m(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$. The $\mathbb{P}_{\overline{\boldsymbol{x}}}(m)$ term weights the distance between $\overline{\boldsymbol{x}}_m$ and $\overline{\boldsymbol{y}}_m$ proportional to the size of $\overline{\boldsymbol{x}}_m$ so that the missing pattern that covers more samples has more weight than the pattern that covers fewer samples of the dataset. The δ_m term accounts for the dissimilarity based on the divergence in the *mp*-distribution.

[†]s need not be a metric. But if s is a metric, then S can also be a metric for appropriate selection of weights.

It is important to understand why we consider δ-terms in the weights γ (in Eq. (5)), and how it relates to our objective. We can instead consider γ^* *without* the δ_m term, i.e., $\gamma^*_m = \mathbb{P}_{\overline{\boldsymbol{x}}}(m)$ for all m, and let $\mathcal{S}^*_{\overline{\boldsymbol{x}}}$ denote the S (given by Eq. (5)) corresponding to γ^*. Now, let us compare $\mathcal{S}^*_{\overline{\boldsymbol{x}}}$ with $S_{\overline{\boldsymbol{x}}}$ to see how (α, β)-closeness captures the desired properties. For instance, when the divergence in *mp*-distributions of $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{y}}$ is zero, i.e., $D_{mis}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) = 0$, we have that $S_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) = \mathcal{S}^*_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$. And when $D_{mis}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) > 0$, it follows that $S_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) > \mathcal{S}^*_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$. That is, the two synthetic datasets that have same similarity score for each missing pattern (with respect to $\overline{\boldsymbol{x}}$), but differ in their *mp*-distributions (from that of $\overline{\boldsymbol{x}}$), the one with higher D_{mis} will be deemed more farther/dissimilar from $\overline{\boldsymbol{x}}$ than the other.

Example. Consider $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{y}}$, both consisting of only two missing pattern m_1 and m_2, such that $\mathbb{P}_{\overline{\boldsymbol{x}}}(m_1) = \mathbb{P}_{\overline{\boldsymbol{y}}}(m_2) = a$ and $\mathbb{P}_{\overline{\boldsymbol{x}}}(m_2) = \mathbb{P}_{\overline{\boldsymbol{y}}}(m_1) = 1 - a$. Now, even if $\mathcal{W}(\overline{\boldsymbol{x}}_{m_1}, \overline{\boldsymbol{y}}_{m_1}) = \mathcal{W}(\overline{\boldsymbol{x}}_{m_2}, \overline{\boldsymbol{y}}_{m_2})$, our consolidated measure is able to capture the differences in the dataset, i.e., $S_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) > \mathcal{S}^*_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$ for all $a \neq 1/2$. In particular, $\mathcal{S}^*_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) = \mathcal{W}(\overline{\boldsymbol{x}}_{m_1}, \overline{\boldsymbol{y}}_{m_1})$ and $S_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) = \sqrt{1 + 2 \cdot |2a - 1|}\mathcal{W}(\overline{\boldsymbol{x}}_{m_1}, \overline{\boldsymbol{y}}_{m_1})$.

Thus, (α, β)-closeness provides a way to assess which of the given two synthetic datasets, $\overline{\boldsymbol{y}}$ and $\overline{\boldsymbol{y}}'$ (with missing values) is closer to the real dataset $\overline{\boldsymbol{x}} \in \mathcal{D}$. α measures the discrepancies in the missing pattern distributions, while β captures the discrepancies in the observable data. Therefore, if $\overline{\boldsymbol{y}}$ and $\overline{\boldsymbol{y}}'$ are respectively (α, β) and (α', β') -closer to $\overline{\boldsymbol{x}}$, then, we say $\overline{\boldsymbol{y}}$ is closer to $\overline{\boldsymbol{x}}$ than $\overline{\boldsymbol{y}}'$ — alternatively, $\overline{\boldsymbol{y}}$ is *better* than $\overline{\boldsymbol{y}}'$ — if $\alpha \leq \alpha'$ and $\beta \leq \beta'$. Thus, using the notion of closeness, we can determine which of the given synthetic datasets, or the method to generate synthetic dataset, is *better*.

V. High Fidelity Synthetic Data Generation

Here, we present our approach and the resulting methods that for a given dataset, $\overline{\boldsymbol{x}} = \{x_1, \ldots, x_n\}$, *generate a synthetic dataset,* $\overline{\boldsymbol{y}} = \{y_1, \ldots, y_n\}$*, that is close to* $\overline{\boldsymbol{x}}$ *in terms of both missing pattern (mp)-distribution and observable data distribution.* We will rely on modeling missing values as missing mechanism, discussed in Section III-B, and the ideas developed in the previous section.

Setting. For synthetic data generation, we consider machine learning based generative methods, such as GANs, VAEs, BNs. We refer to them as *data generation methods (DGMs)*. Note that, as discussed in Section III-C, there are variety of ways to guarantee varying levels of privacy when using DGMs. Our focus here is to develop an *mp*-distribution preserving data generation approach using any of the existing DGMs.

Before presenting our approach we share a **key observation**, which is instrumental in solving the problem in practical settings: the number, ℓ, of missing patterns (MPs) in a $\overline{\boldsymbol{x}}$ is much smaller than the total number of possible MPs, i.e., 2^d, for d-dimensional data. In addition, a much smaller number of MPs cover most of the samples in datasets. For instance, in about 16 of 19 real datasets (from Kaggle and KEEL repositories), 2 MPs account for more than 80% of the samples. Consider the Brain dataset [45], which contains

101 attributes, thus allowing up to $2^{101}-1$ potential patterns. Nevertheless, it contains only 96 patterns in total, with two predominant patterns, while the majority of the other patterns include fewer than ten records each. This means that we can extract a lot of value from data by considering a few additional MPs besides complete-samples. Therefore, we primarily focus on the setting where $\ell << 2^d$, though our methods are applicable in more general settings.

Solution Overview

Consider the joint distribution $\mathbb{P}(X^o, M)$. Recall (from Section III) that X^o is the random variable for complete data samples (i.e., without any missing values); also, note that for (X^o, M) distributed according to $\mathbb{P}(X^o, M)$, $X = X^o \oplus M$, where $\oplus$-operation is such that $X_j = \texttt{NA}$ if $M_j = 1$ otherwise $X_j = X^o_j$. Therefore, it follows that

$$\mathbb{P}(X) = \mathbb{P}(X^o, M) = \mathbb{P}(X^o|M)\,\mathbb{P}(M). \tag{6}$$

The above equation gives two ways to approach the problem. Approach #1: Learn the joint distribution of the observable data and *mp*-distribution. Approach #2: Learn the *mp*-distribution separately and then learn the conditional distribution of the observable data, that is, only under a fixed missing pattern, i.e., $\mathbb{P}(X^o|M)$. Note that in the second case, when learning $\mathbb{P}(X^o|M)$, we are only concerned with learning the observable part for that pattern, and hence, disregard all the missing features/columns.

Let us first discuss Approach #2: It is well suited for MCAR setting, where $\mathbb{P}(X^o, M) = \mathbb{P}(X^o)\,\mathbb{P}(M)$ (i.e., the missingness is independent of the data), and non-MCAR settings (i.e., $\mathbb{P}(X^o, M) \neq \mathbb{P}(X^o)\,\mathbb{P}(M)$) with sufficient support (i.e., number of samples) for hott partitions (explained shortly). Now, for a given dataset, $\overline{\boldsymbol{x}}$, which is sampled according to $\mathbb{P}(X)$, the *mp*-distribution can be learned as follows:

$$\mathbb{P}_{\overline{\boldsymbol{x}}}(m) = |\overline{\boldsymbol{x}}_m|/|\overline{\boldsymbol{x}}| \tag{7}$$

for every $m \in \mathcal{MP}$, where $\overline{\boldsymbol{x}}_m$ is the hott partition (samples from $\overline{\boldsymbol{x}}$ with MP as m). Since in practical settings (as discussed above) the number of actual MPs in $\overline{\boldsymbol{x}}$ are much smaller, the above method is an efficient way to learn a very good approximation of $\mathbb{P}(M)$. When the missing mechanism is MCAR, we use the method *f/pRand* (Section V-A) that learns the two distributions independently and uses them to generate synthetic data that is close in terms of observable and missing data distributions. For non-MCAR setting, we propose HottGEN (in Section V-B), wherein the process of learning of the *mp*-distribution remains the same as in in f/pRand, however, the learning of the conditional observable data distribution is a little more involved: here, multiple generators are learned, one for each hott partition, $\overline{\boldsymbol{x}}_m$ (all samples from $\overline{\boldsymbol{x}}$ with the same missing pattern m).

To realize Approach #1, i.e., learn $\mathbb{P}(X^o, M)$, we propose MergeGEN, wherein a DGM is trained over a dataset in which MPs and samples are combined as one dataset, i.e., *merged*. Once we can generate samples, $(z_1, m_1), \ldots, (z_n, m_n)$, from $\mathbb{P}(X^o, M)$, we can generate samples, $y_1, \ldots, y_n$ corresponding to the distribution $\mathbb{P}(X)$ by creating missing values in each z_i as per its missing pattern m_i. MergeGEN has the added benefit that it works even when the number of MPs is not small.

Approach #1 and Approach #2 each have their benefits and limitations: Approach#2 is straightforward and efficient but can perform poorly when the number of MPs in the data is large. Conversely, Approach #1 is computationally more demanding for DGMs such as GANs and VAEs but can handle a larger number of MPs. This capability is beneficial when hott partitions lack sufficient samples for training the generator. To address these issues, we combine the two approaches into a comprehensive solution, which we refer to as *HottGEN+*. The following sections present the details.

A. Learning over Complete-Samples with Independent Missing Patterns (fRand, pRand)

Here, we learn a generator, G_{m^*}, via the given DGM, by using the set of complete samples, $\overline{\boldsymbol{x}}_{m^*}$, from the given dataset, $\overline{\boldsymbol{x}}$, where $m^* = \mathbf{0}$. Thus, $G_{m^*} \leftarrow \texttt{DGM}(\overline{\boldsymbol{x}}_{m^*})$. Note that G_{m^*} generates complete d-dimensional samples. To learn the *mp*-distribution ($\mathbb{P}(M)$), we use Eq. (7) and calculate the proportion of each missing pattern in $\overline{\boldsymbol{x}}$: $\mathbb{P}_{\overline{\boldsymbol{x}}}(M)$ denotes the learned *mp*-distribution. Indeed, for any m that is not present in $\overline{\boldsymbol{x}}$, $\mathbb{P}_{\overline{\boldsymbol{x}}}(m) = 0$.

To generate a synthetic dataset of size N, set $\overline{\boldsymbol{y}} = \{\}$. For every m with $\mathbb{P}_{\overline{\boldsymbol{x}}}(m) > 0$, generate $(N \cdot \mathbb{P}_{\overline{\boldsymbol{x}}}(m))$-many samples from G_{m^*} and create missing values in them according to the missing pattern m (i.e., replace the j-th feature value by $\texttt{NA}$ for $m_j = 1$ for every j), and add them to $\overline{\boldsymbol{y}}$. We call this method pRand, where 'p' denotes pattern-level.

Another way is to consider missingness at the feature level: here, missing values are created independently for each feature in the generated samples. This is done by tossing a coin for each sample and each feature j, which gives Heads (corresponds to creating missing value) with probability equal to the missing rate of the feature j. Thus, the missing rate $\mathbb{P}(M_j = 1)$ decides how many samples have values missing in feature j, where $M = (M_1, \ldots, M_j, \ldots, M_d)$ is for the missing pattern and M_j is the random variable for missing value for feature j. The missing rate of the feature j can be estimated as: $\mathbb{P}_{\overline{\boldsymbol{x}}}(m_j = 1) = |\{x \text{ in } \overline{\boldsymbol{x}} \mid x_j = \texttt{NA}\}|/|\overline{\boldsymbol{x}}|$. This method is called fRand, where 'f' denotes feature-level.

Remarks: One drawback of fRand (compared to pRand) is that it produces a larger number of missing patterns, and thus, diverges from the original *mp*-distribution whenever the missingness in two features, j and j', (correspondingly M_j and $M_{j'}$) are not independent. Hence, as the number of such pairs of features increases, so does the divergence in the *mp*-distribution of the dataset produced by fRand (we will see this in the Evaluation, Section VI-C).

The independence assumption (about features' missingness) rarely holds in practice. Even the natural setting where the number of MPs in the dataset is small implies a very high correlation in the missingness of the features. In addition, pRand and fRand face further data quality loss when the missingness is not MCAR, i.e., $\mathbb{P}(X \mid M = m^*)$ is (very) different from $\mathbb{P}(X)$. This is because G_{m^*} produces samples from $\mathbb{P}_{\overline{\boldsymbol{x}}}(X \mid M = m^*)$ instead of $\mathbb{P}_{\overline{\boldsymbol{x}}}(X)$. Therefore, to deal with such non-MCAR setting, we next present HottGEN and

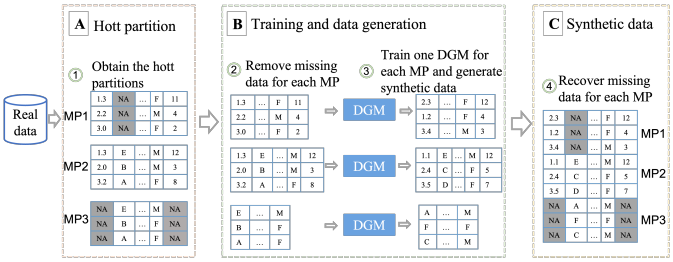

Fig. 1: HottGEN. The grey color depicts missing values.

MergeGEN, two powerful methods to generate synthetic data while preserving utility and *mp*-distribution.

B. Learning over Homogeneous Pattern Partitions (HottGEN)

This approach uses a collection of generators, learned over *hott partition* of the dataset, $\overline{\boldsymbol{x}}$. The hott partitions correspond to different missing patterns in $\overline{\boldsymbol{x}}$, where each missing pattern m in $\overline{\boldsymbol{x}}$, the corresponding hott partition, $\overline{\boldsymbol{x}}_m$ only consists of the samples with the missing pattern m (defined below).

Definition 2 (Hott partition): For any given dataset $\overline{\boldsymbol{x}}$, let $\mathcal{M} = \{m^1, \ldots, m^k\} = \texttt{miss-patt}(\overline{\boldsymbol{x}})$ be the set of missing patterns in $\overline{\boldsymbol{x}}$. Then, $\overline{\boldsymbol{x}}_{m^1}, \ldots, \overline{\boldsymbol{x}}_{m^k}$ make the homogeneous pattern (*hott*) partition of $\overline{\boldsymbol{x}}$ if $\overline{\boldsymbol{x}} = \bigcup_{m \in \mathcal{M}} \overline{\boldsymbol{x}}_m$ and each $\overline{\boldsymbol{x}}_{m^j}$ contains the samples with missing pattern m^j.

Algorithm 1 gives the detailed HottGEN approach to generate synthetic data using hott partitions, while Figure 1 gives a pictorial overview of the algorithm. Below, we give an overview of the approach.

HottGEN begins by first obtaining the hott partition of $\overline{\boldsymbol{x}}$ (line-1). Since all the samples in $\overline{\boldsymbol{x}}_{m^j}$ (i.e., hott partition) have the same missing pattern, we remove all of its columns with missing values (line-5) without affecting the observable data. Moreover, we only consider the partitions that have a minimum support τ (line-4): this ensures that we have sufficient data to train the GAN (line-6).

Once the generators are learned, we use them to generate a synthetic dataset of size N (line 10-15). For each pattern m^j with at least τ-support (i.e., $|\overline{\boldsymbol{x}}_m| \geq \tau$), we calculate the proportionally appropriate number, n_j, of samples with missing pattern m^j. Next, n_j samples are generated using the generator G_{m^j} (line-12), followed by the addition of the missing columns, and the collection of all the generated samples with MPs (line 12-15) to produce the synthetic dataset.

Remarks: We note that HottGEN only considers the hott partitions with τ-support; thus, in the settings of our interest (discussed in the beginning of Section V), this result in a little higher than zero divergence in *mp*-distribution. That is, if λ is the proportion of the samples covering all the hott partitions lacking τ-support, then $D_{mis}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}) \leq 2\lambda$ (see Section A). In Section VI, we will see that HottGEN is in fact one of the best performing methods to generate synthetic data.

C. Learning Data and Missingness Together (MergeGEN)

Here, we propose a method to learn $\mathbb{P}(X^o, M)$, i.e., the observable data distribution together with the *mp*-distribution.

Algorithm 1 HottGEN

Input: $\overline{\boldsymbol{x}} = \{x_1, \ldots, x_n\}$ (real dataset) and τ (min support)
Output: $\overline{\boldsymbol{y}} = \{y_1, \ldots, y_n\}$ (synthetic dataset)

Build HottGEN:
//Create hott partitions of the data
1: $\{\overline{\boldsymbol{x}}_{m^1}, \ldots, \overline{\boldsymbol{x}}_{m^k}\} = \texttt{hott-partition}(\overline{\boldsymbol{x}})$
// Use DGMs to learn the synthetic data generator over
// the hott partitions with at least τ-support
2: $\mathcal{M}_\tau = \emptyset$ // will contain the MPs with τ-support
3: **for** $j = 1$ to k **do**
4: **if** $\texttt{size}(\overline{\boldsymbol{x}}_{m^j}) \geq \tau$ **then**
5: $\overline{\boldsymbol{z}} = \texttt{remove-missing-cols}(\overline{\boldsymbol{x}}_{m^j})$
6: $G_{m^j} = \texttt{DGM}(\overline{\boldsymbol{z}})$
7: $\mathcal{M}_\tau = \mathcal{M}_\tau \cup \{m^j\}$
8: **end if**
9: **end for**

Generate synthetic dataset of size N via HottGEN
10: **for** m in $\mathcal{M}_\tau$ **do**
// Compute the number of records
// to be sampled as per $\mathbb{P}_{\overline{\boldsymbol{x}}}(m)$
11: $n_j = N \times \frac{\texttt{size}(\overline{\boldsymbol{x}}_m)}{\sum_{p \in \mathcal{M}_\tau} \texttt{size}(\overline{\boldsymbol{x}}_p)}$
// Generate n_j synthetic data samples
// using the generator G_m
12: $\overline{\boldsymbol{z}}' = \{z_1, \ldots, z_{n_j}\}$ s.t. $z_i \leftarrow G_m$ for every i
13: $\overline{\boldsymbol{y}}_m = \texttt{add-missing-cols}(\overline{\boldsymbol{z}}', m)$
14: **end for**
15: $\overline{\boldsymbol{y}} = \bigcup_{m \in \mathcal{M}_\tau} \overline{\boldsymbol{y}}_m$ and shuffle the indices of the samples in $\overline{\boldsymbol{y}}$

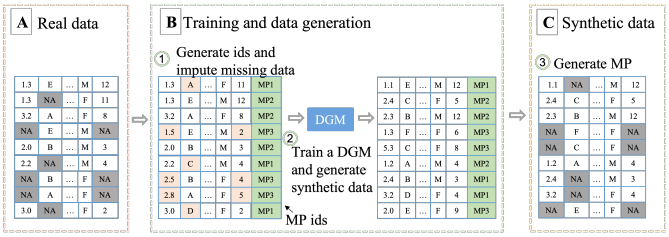

Fig. 2: MergeGEN. Grey, white, and orange colors respectively denote missing, non-missing, and imputed values, while green denotes missing pattern (MP) ids.

Algorithm 2 gives the details while Figure 2 gives a pictorial overview. Below, we give an overview of the approach.

MergeGEN begins by creating categorical ids for each missing pattern in the given dataset ($\overline{\boldsymbol{x}}$) — we refer to these ids as *missing pattern ids* or *MP ids*. We use categorical data type for the MP ids instead of integers (or ordinals) to prevent the DGM, such as GANs, from making use of the geometric or other numeric properties that are not related to MPs. Using the MP ids, we create (hash) maps for mapping MP ids to missing patterns and vice versa (line 1-6). We then use the pattern-to-id ($p2i$) mapping to generate MP ids (ID_i) for each sample x_i in $\overline{\boldsymbol{x}}$ (line 7-9). Since most DGMs cannot learn the generator using the missing values, we impute all the missing values in $\overline{\boldsymbol{x}}$, and add the MP ids as an additional feature to the imputed

Algorithm 2 MergeGEN

Input: $\overline{\boldsymbol{x}} = \{x_1, \ldots, x_n\}$ (real dataset), N (size of synthetic data)
Output: $\overline{\boldsymbol{y}} = \{y_1, \ldots, y_N\}$ (synthetic dataset)

Setup and training:
//Create one-to-one mapping from missing patterns
//to categorical ids
1: $\mathcal{M} = \texttt{miss-patt}(\overline{\boldsymbol{x}})$ //the set of MPs in $\overline{\boldsymbol{x}}$
2: Initialize $p2i$ and $i2p$ (hash maps) and set $id = 1$
3: **for** m in $\mathcal{M}$ **do**
4: $sid = \texttt{concat}(\text{“MP”}, \texttt{str}(id))$ //int to string
5: Set $p2i(m) = sid,\ i2p(sid) = m$, and $id = id + 1$
6: **end for**
//Generate MP ids for all samples in $\overline{\boldsymbol{x}}$
7: **for** $i = 1, 2, \ldots, n$ **do**
8: $ID_i = p2i(\texttt{miss-patt}(x_i))$
9: **end for**
//Learn the synthetic data generator via a DGM
10: $\overline{\boldsymbol{z}} = \texttt{impute-missing-values}(\overline{\boldsymbol{x}})$
11: $\overline{\boldsymbol{z}}' = \texttt{add-feature}(\overline{\boldsymbol{z}}, ID)$ //$z'_i = (z_i, ID_i)$
12: $G = \texttt{DGM}(\overline{\boldsymbol{z}}')$

Generating synthetic dataset:
13: $\overline{\boldsymbol{w}} = \{w_1, \ldots, w_N\}$, where $w_j \leftarrow G$ for every j
14: $\overline{\boldsymbol{w}}' = \texttt{create-missing-patterns}(\overline{\boldsymbol{w}}, i2p)$
15: $\overline{\boldsymbol{y}} = \texttt{remove-ID-feature}(\overline{\boldsymbol{w}}')$ and shuffle indices of the samples in $\overline{\boldsymbol{y}}$

$\overline{\boldsymbol{x}}$ to obtain the processed dataset $\overline{\boldsymbol{z}}'$ (line 10-11). Now we use the given DGM to learn the synthetic data generator, G, over the processed dataset (line 12).

To generate synthetic data (of size e.g., $N = n$), we use the generator to produce N samples (line-13), and create missing patterns as per the MP id in each of the generated samples (line-14). Finally, we remove the MP id feature (i.e., ID) to produce the synthetic dataset with missing values (line-15).

Remarks: One way MergeGEN is more powerful than the prior methods is that it can learn the complex interactions, correlations, and dependencies among features as well as features and missing patterns; furthermore, it can handle the settings with larger number of MPs, where the other method may perform poorly. However, it does come at the cost of a significant computational overhead as the DGM uses the whole dataset to learn the generator; this is particularly true for methods such as GANs.

D. HottGEN+: A Hybrid Approach and a Complete Solution

In the previous sections, we proposed methods (HottGEN, pRand, and MergeGEN) for synthetic data generation while preserving observable and *mp*-distributions, and also briefly discussed their limitations and strengths. Here, we present a way to combine HottGEN with other methods to give a hybrid approach, where the limitation of one method are addressed by the other, greatly increasing the quality of the generated data. Here, we consider HottGEN and non-MCAR setting because for MCAR setting, pRand suffices (see Section A) as long as $\overline{\boldsymbol{x}}_{m^*}$ (the set of complete samples) is large enough to learn the distribution, otherwise MergeGEN should be used for MCAR setting. Thus, this approach is named *HottGEN+*.

Let us look at one way to quantifiably improve the quality of the synthetic data generated by HottGEN. Let $\mathcal{M}$ and $\mathcal{M}_\tau$ respectively be the missing patterns present in $\overline{\boldsymbol{x}}$ and $\overline{\boldsymbol{y}}$, where $\overline{\boldsymbol{y}}$ is generated by HottGEN, and let $|\overline{\boldsymbol{x}}_m| = |\overline{\boldsymbol{y}}_m|$ for each $m \in \mathcal{M}_\tau$ (recall that $\mathcal{M}_\tau \subseteq \mathcal{M}$ conisits of MPs that have τ-support in $\overline{\boldsymbol{x}}$). We would like to generate additional synthetic data samples (corresponding to the *left-over patterns*, i.e., $\overline{\boldsymbol{z}} = \cup_{m \in \mathcal{M} \setminus \mathcal{M}_\tau} \overline{\boldsymbol{x}}_m$) which will be included in $\overline{\boldsymbol{y}}$ to boost its quality in terms of (α, β)-closeness.

We describe the basic idea via a naïve methodology, called *HottGEN+Naïve*, which augments the HottGEN's generated data with additional samples and provably improves the data quality. To do this, we compute the average, μ_m, of all the samples in each $\overline{\boldsymbol{x}}_m$ for every $m \in \mathcal{M} \setminus \mathcal{M}_\tau$, i.e., $\mu_m = (\sum_{x \text{ in } \overline{\boldsymbol{x}}_m} x)/|\overline{\boldsymbol{x}}_m|$. Then, for each $m \in \mathcal{M} \setminus \mathcal{M}_\tau$: we add $|\overline{\boldsymbol{x}}_m|$-many μ_m duplicates to $\overline{\boldsymbol{y}}$. This produces the synthetic dataset $\overline{\boldsymbol{y}}^+$. Now, for $\overline{\boldsymbol{y}}^+$, $D_{miss}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}^+) = 0$, which means we have been able to preserve the *mp*-distribution in the synthetically generated dataset. Thus, $D_{miss}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}^+) < D_{miss}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$ when $\mathcal{M} \setminus \mathcal{M}_\tau$ is not empty. Furthermore, $S_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}^+) \leq S_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$ (see Section A for details). Thus, *HottGEN+Naïve is better than HottGEN if $\mathcal{M} \setminus \mathcal{M}_\tau$ is not empty*, i.e., there are some missing patterns (not all) that lack τ-support.

HottGEN+: Now instead of the naïve methodology, we can generate the required samples (corresponding to the left-over data) using pRand or MergeGEN. However, in the case of MergeGEN, there is a small change in the learning phase, where instead of the whole dataset, $\overline{\boldsymbol{x}}$, we will only use $\overline{\boldsymbol{z}} = \cup_{m \in \mathcal{M} \setminus \mathcal{M}_\tau} \overline{\boldsymbol{x}}_m$ as the input data.

Remarks: It is quite obvious that, for most practical setting, MergeGEN (and pRand) will produce samples (for the left-over patterns) that will outperform the naïve methodology. Hence, the synthetic datasets, produced by HottGEN+pRand or Hott+MergeGEN will be closer (Definition 1) to $\overline{\boldsymbol{x}}$ than the one generated by the HottGEN+Naïve.

E. Discussion and Guidelines

This section summarizes the insights and guidelines for an effective application of our proposed approaches and how to decide which one is appropriate for a given setting. Firstly, consider MCAR setting. If the set of complete-samples, i.e., $\overline{\boldsymbol{x}}_{m^*}$, has sufficient support, pRand is the most computationally efficient method to generates datasets that are as close to $\overline{\boldsymbol{x}}$ as the dataset generated by any other method. This is because, under MCAR, $\mathbb{P}(X \mid M = m^*) = \mathbb{P}(X^o)$ and can learned well by a good DGM (e.g., GAN or BN) using $\overline{\boldsymbol{x}}_{m^*}$. In this case, using another methods will incur a higher computational cost without any additional improvement.

Let us next consider the more realistic setting, i.e., when the missing mechanism is *not* MCAR: namely, the missing mechanism is either MAR or MNAR or a combination of them and MCAR with different features possibly missing under different mechanisms. Under the non-MCAR setting, pRand is bound to perform extremely poorly, especially when

the distributional mass (of $\mathbb{P}(M)$) is not overwhelmingly concentrated for the missing pattern m^*. That is, $\mathbb{P}(M = m^*)$, is not large, e.g., $\mathbb{P}(M = m^*) < 0.3$.

In non-MCAR settings, HottGEN, MergeGEN, and HottGEN+ are a much better fit. HottGEN is preferred when all or most of the hott partitions have sufficient support or most of the mass of $\mathbb{P}(M)$ is concentrated in some k hott partitions that have sufficient support so that DGM can learn a generator that have satisfactory performance. In case, the k hott partitions have sufficient data and rest of the hott partitions together provide large enough dataset, HottGEN+ can be used to improve the quality of the generated data.

However, if all the hott partitions lack sufficient support, then pRand and HottGEN (and even HottGEN+) will perform poorly. Here, using MergeGEN — which makes use of all the data and not any single hott partition – obviates the data insufficiency problem and can give a generator that produces better quality synthetic data, but it does incur a higher computational cost for neural network based methods.

VI. Empirical Evaluation

We carry out an extensive empirical evaluation measuring performance using a variety of different metrics over both carefully fabricated and real world datasets; the evaluation considered a range of different missing data settings, our approaches augmented with various data generation methods, for example, HottGAN uses a GAN-based generator, while HottBN relies on a Bayesian Network, and HottVAE employs Variational Autoencoders within the HottGen framework, which we compare against other baseline approaches. Key observations from the evaluation include:

- In all scenarios involving missing data, synthetic data that maintains both observable and missing data distributions is of higher quality compared to synthetic data produced using complete-case analysis (i.e., deletion) or the impute-then-generate approach.
- Our methods significantly outperform other methods (e.g., MisGAN [23] and Bayesian Networks [46]) to generate synthetic data with missingness.
- In particular, when missing data settings is non-MCAR and missing patterns are many, HottGAN and MergeGAN outperform other methods, the baselines, as well as fRand and pRand.
- The hybrid methodology, e.g., Hott+MergeGAN, supports fine-tuning the quality vs. efficiency tradeoff for complex missing mechanisms: it improves data quality (compared to HottGAN) and reduces the computation cost (compared to MergeGAN) (Section VI-B).
- Bayesian Networks (BN) are particularly well suited to our generation methodology. Indeed, HottBN and MergeBN perform the best of all of the methods (Tables I and II).
- $S_{\overline{\boldsymbol{x}}}$, i.e., the proposed weighted average Wasserstein distance based similarity measure very well tracks the comparative performance of the synthetic data generation methods.

Assessment Criteria. Several subjective and objective measures are used to measure the quality of the synthetic data produced. First, **t-SNE** (t-distributed stochastic neighbor embedding in 2D) analysis [47] to visually assess the fidelity of the synthetic datasets across different missing patterns. For discrete features, the χ^2-test validates the hypothesis that the real and synthetic data are from the same (i.e., sufficiently close) distribution(s). For numeric features, the relative error of mean (*REM*) and relative error of standard deviation (*RESD*) (of real and synthetic datasets) provide a quantitative measurement—they are respectively reported as the average of REM and RESD. In addition, we assess the overall quality of synthetic data using $S_{\overline{\boldsymbol{x}}}$ (Eq. (5)), i.e., *weighted Wasserstein distance* across missing patterns (note that since sliced-Wasserstein distance is computed using multiple projections, only numeric features are used for this). The quality of the generated missing pattern distribution is measured via the *mp-distribution divergence* (Eq. (2)), i.e., D_{mis}.

Other methods and Baselines. We compare our methods with two baselines, i.e., the *Deletion* method (where only complete samples are used from the real data) and the *Imputation* method (where missing values in real data are imputed by 3-Nearest Neighbors algorithm before using the data). In addition, we compare two other methods that can generate synthetic data with missing values: *MisGAN* [23], which consists of two different GANs that work in tandem to learn observable and missing data distributions; *Bayesian Network (BN)* [46] and *CTGAN* [31], which learns missing data distribution by treating it as a special value.

All our methods are implemented using Python 3.6.0, PyTorch 1.8.0, and CTGAN [31] (which is the state-of-the-art tabular data generation model and can handle mixed data types) with its default configuration. For Bayesian Network (BN), we used bnlearn [48]; for MisGAN, we used the authors' provided implementation [23] with the given configuration, except we omitted Sigmoid function in the output layer to avoid synthetic data getting mapped between 0 and 1. The evaluation was done on a Windows Server 2012 R2 with 128G RAM and Intel(R) Xeon(R) CPU E5-2640.

A. Evaluation over Fabricated Data

We use two types of fabricated datasets, generated using the Gaussian mixture model and missing at random (MAR) mechanism (i.e., $\mathbb{P}(X, M) = \mathbb{P}(M \mid X)\mathbb{P}(X) \neq \mathbb{P}(M)\mathbb{P}(X)$). Following is their overview (and the details are given in Appendix. B).

Gauss 1 datasets allow us to evaluate all the methods in a simple but crucial setting. Each Gauss 1 dataset consists of 3 features and 10000 records, with two correlated and one independent feature. The missing mechanism depends on feature 1 and creates missing values in feature 2. For a given quantile q: it replaces the value of feature 2 by `NA` (probabilistically by flipping a coin) if feature 1's value is below q-th quantile. Note that the randomization, introduced via the coin flip, is crucial to mimic real-world situations where the missingness is not completely deterministic. We use $q = 0.2, 0.4, 0.6, 0.8$ to obtain 5 different Gauss 1 datasets, each with different missingness correlations and extent.

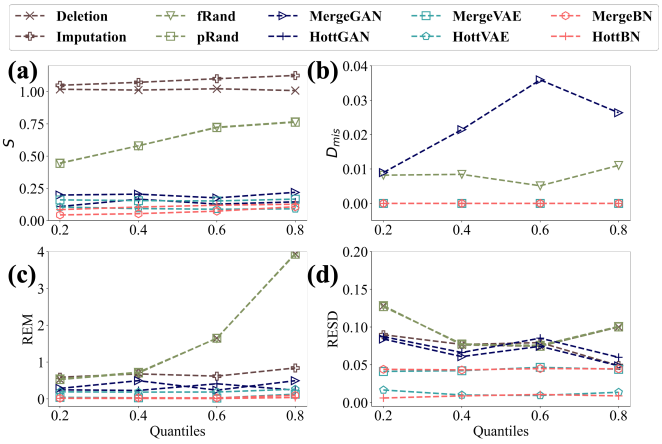

Fig. 3: Synthetic data quality, Gauss 1. Each quantile refers to a different dataset, $\overline{\boldsymbol{x}}$, with different MAR-missingness. REMS and RESD resp. give relative error of mean and standard deviation of the synthetic dataset, $\overline{\boldsymbol{y}}$, and $S = S_{\overline{\boldsymbol{x}}}(\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}})$.

Gauss 2 (compared to Gauss 1) has more missing patterns (MPs) and a complex *mp*-distribution. With Gauss 2, we simulate a real world like correlation among multiple features. Each Gauss 2 dataset consists of 6 features and 25000 records, where except for one feature, all others are correlated (for different coefficient values). The missing values are created in 4 of its 6 features, each with a different specification (i.e., quantile value) of the missing mechanism. This mechanism works exactly like the one for Gauss 1 except for one difference: it probabilistically depends on two features; it depends on feature 1 or feature 2 (depending upon a fair coin flip) to create missing values, just like for Gauss 1. For the missing mechanism, we choose features 3, 4, 5, and 6 with a different quantile for each: the quantiles $0.2, 0.4, 0.6$, and 0.8 for features 3, 4, 5, and 6, respectively. Thus, these datasets help evaluate the performance of the *mp*-distribution preserving synthetic data generation methods under MAR-missingness. For the evaluation, we independently sample 5 datasets for each type and specification.

Results. On all fabricated datasets, our methods outperform all the baselines and the other methods on almost all the evaluation metrics (Fig. 3, Table I, and Fig. C.1). Note that the results for Bayesian Network (BN) and MisGAN are extremely bad and have been omitted from Fig. 3 so that the differences among the methods are observable. For instance, for $q = 0.2$, BN achieved REM=34.4, RES=5.7, $S_{\overline{\boldsymbol{x}}}$=0.67, and D_{mis}=0.68, while MisGAN achieved REM=1.5, RES=0.67, $S_{\overline{\boldsymbol{x}}}$=1.6, and D_{mis}=0.74. Furthermore, since Deletion and Imputation methods only generate complete-samples, they perform poorly on D_{mis} and $S_{\overline{\boldsymbol{x}}}$; thus, we omit their results for these metrics from Fig. 3 as well.

In particular, on Gauss 1, HottBN demonstrates the most favorable performance, followed by MergeBN, HottVAE, and HottGAN, as illustrated in Fig. 3. Although Imputation is close to MergeGAN in performance, Imputation drawback, of course, is its inability to preserve missing data distribution in the synthetic data. Deletion, fRand, and pRand performed poorly—still better than BN and MisGAN—because missing data is not MCAR.

TABLE I: Synthetic data quality, Gauss 2. For each method, Score gives (out of 5) the number of metrics as per which the method is among the top-2; blue color for top-1 and lightblue for top-2. While other metrics are the same as Fig. 3. PCD is given in the text.

Methods	REM	RESD	PCD	D_{mis}	$S_{\overline{\boldsymbol{x}}}$	Score
Deletion	0.79	0.16	0.92	-	-	0/5
Imputation	0.15	0.12	0.52	-	-	2/5
MisGAN [23]	1.21	0.81	2.15	1.2	1.9	0/5
BN [46]	31.53	16.53	2.30	1.01	2.2	0/5
fRand	0.79	0.17	0.94	0.83	1.35	0/5
pRand	0.79	0.17	0.91	0	0.7	1/5
MergeGAN	0.14	0.08	0.52	0.01	0.29	1/5
MergeVAE	0.03	0.09	1.54	0	0.17	1/5
MergeBN	0.03	0.10	0.27	0	0.05	4/5
HottGAN	0.14	0.07	0.82	0	0.17	2/5
HottVAE	0.06	0.01	1.28	0	0.15	2/5
HottBN	0.01	0.01	0.30	0	0.09	5/5

On Gauss 2, the trends in the results are also similar to what we saw above (see Table I). For Gauss 2, we also computed the error in estimating the original correlation matrix—referred to as **PCD** [49]: it is computed as the Frobenius norm of the difference of the original and estimated correlation matrices. Furthermore, the t-SNE plots (Fig. C.1 give additional confirmation for the results (depicted in Fig. 3 and Table I).

Thus, the results show that when the missingness is MAR, generating synthetic data while preserving *mp*-distribution leads to higher quality synthetic data. Furthermore, under such missingness, HottBN and MergeBN are better options compared to fRand and pRand. Lastly, we note that under MAR missingness, MisGAN and Bayesian Network both are bad choices to generate synthetic data with missingness.

B. Evaluation of Tradeoffs over "Complex"-Missingness

Here, we demonstrate how HottGAN+ (similary for HottVAE+ and HottBN+) improves over both HottGAN (HottVAE and HottBN) and MergeGAN (MergeVAE and MergeBN) under non-MCAR missingness when some missing patterns lack sufficient samples. The evaluation is carried out over a carefully fabricated dataset, Gauss 3, sampled from a multivariate Gaussian distribution; the missing values are created by different MCAR, MAR, and MNAR mechanisms. **Gauss 3** consists of 50,000 records and 6 features with different degrees of correlation coefficients. The missing data is produced by three different missing mechanisms, encompassing MCAR, MAR, and MNAR. As a result, Gauss 3 contains 31 missing patterns. We select the 21 largest missing patterns. The largest missing pattern contains 17977 records and consists of 36.7% of the total records, while the smallest one (among the 21 missing patterns) consists of 0.4% of the total records. The details of the instantiation of each type of missiness are provided in B.

Results: Figure 4 shows how different versions of HottGEN+ compare in terms of data quality for different volumes of data being processed by HottGEN. For the evaluation, we train HottGAN over the top-k patterns (i.e., the k hott partitions with the most support) and generate the corresponding synthetic data. For rest of the patterns, we generate synthetic data using Naïve method (HottGAN+Naïve), pRand

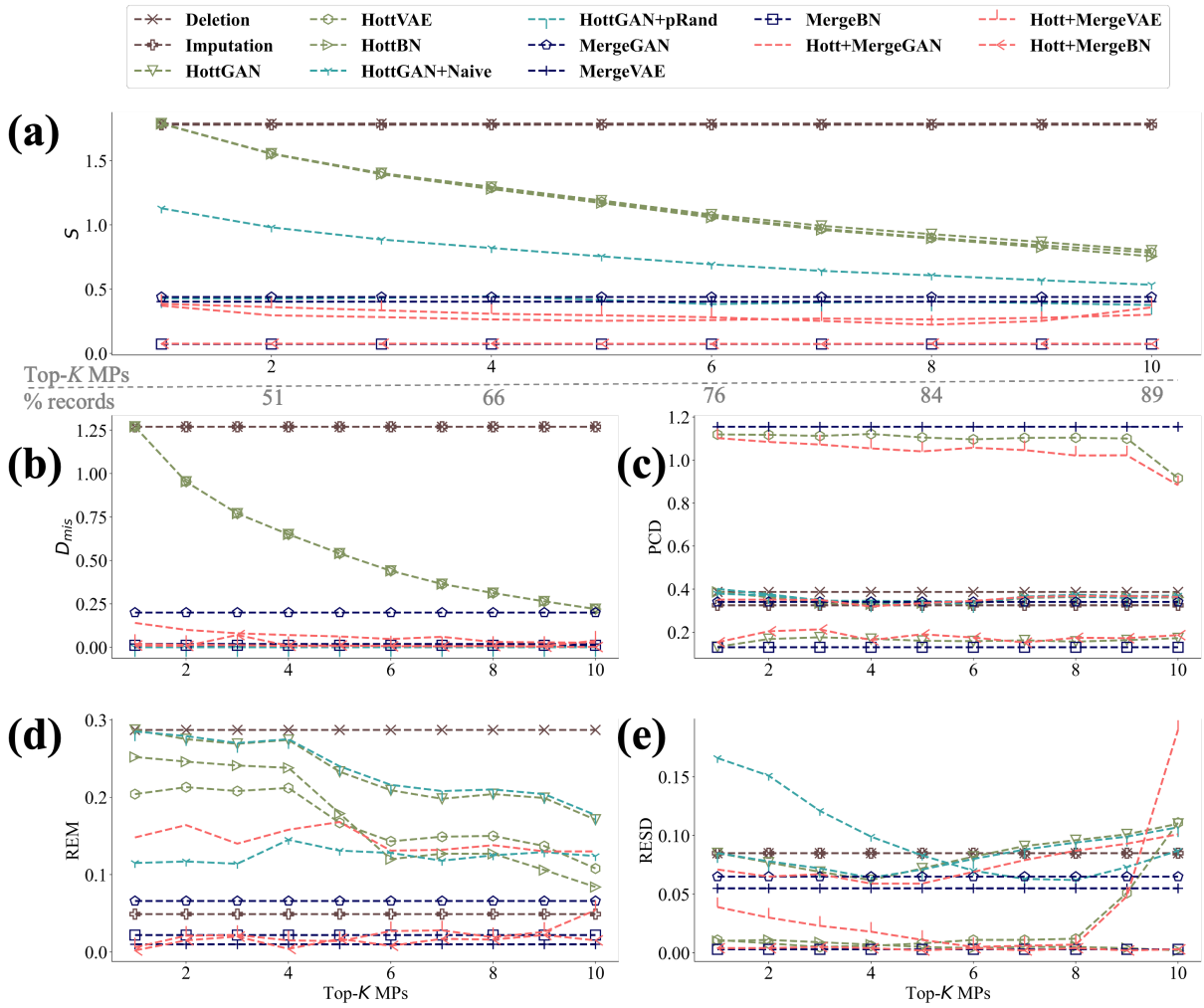

Fig. 4: Synthetic data quality, Gauss 3. (a) horizontal axis gives two values for each tick, the top one gives top-k, i.e., k hott partitions with the highest support; the bottom one gives the percentage size of top-k. (b)-(e) horizontal axis gives top-k MPs.

(HottGAN+pRand), and MergeGEN (Hott+MergeGAN). Additionally, we train MergeGAN on the entire dataset as a baseline. Moreover, we implement the VAE-based (including HottVAE, MergeVAE, and Hott+MergeVAE) and BN-based (including HottBN, MergeBN, and Hott+MergeBN) approaches following the same methodology.

From Figure 4, Hott+MergeBN produces the highest quality datasets consistently, followed by MergeBN and Hott+MergeVAE. Notably, in comparison to HottGAN and MergeGAN, Hott+MergeGAN generates synthetic data of better quality. It is noteworthy that Hott+MergeGAN achieves this with only approximately $\frac{1}{3}$ of the training time required by MergeGAN when $k = 10$. Similarly, the VAE-based and BN-based versions reduce the training time by around $\frac{1}{3}$.

When $k \geq 10$, HottGAN+pRand and Hott+MergeGAN produce datasets of similar quality, which is not surprising since around 90% of the data is generated by HottGAN. Moreover, during the evaluation, we observe that adding δ_m penalty to the weights enables us to account for divergence in *mp*-distributions while assessing similarity. This is demonstrated by the increased value of $S_{\overline{\boldsymbol{x}}}$ in comparison to $\mathcal{S}^*_{\overline{\boldsymbol{x}}}$.

C. Evaluation over Real Data with Missingness

We use two real world datasets: *Price* and *Brain* (Appendix D, Table III provides the details.) Both datasets have missing values, which are either MAR or MNAR as confirmed by Little's test [50], [51] for MCAR (extremely low p-value=0). The missing rates for Price [52] range from 0.8% to 49.6%, and that of Brain [45], from 0.002% to 40.98%. Brain has 53 missing patterns, out of which we select top-2 (covering 93.3% of the samples) as the rest consisted of a few samples.

Here, we additionally use the **downstream task** of classification to characterize the performance of all the methods. For each dataset, we identify a categorical feature that is correlated with the missing patterns and appropriately convert it to a binary feature, and use it as the target feature. We then use *CART* [53], Logistic Regression (LR), and Linear Support Vector Machine (*SVM*) [54] to build three different classifiers. *To measure the classification performance*, we use the standard method of Training on Real and Testing on Synthetic data (TRTS) as well as Training on Synthetic and Testing on Real (TSTR) [55], and then compute the area under ROC curve (AUROC) [56]. In order to assess the comparative performance of synthetic data generation methods, we measure these scores

TABLE II: Quality of the synthetic data constructed from real data. Data Quality Measures and Scores are the same as in Table. I. Downstream Task reports the pAUROC (explained in the text).

	Price								Brain							
	Data Quality Measures				Downstream Task			Score	Data Quality Measures				Downstream Tasks			Score
Methods	REM	RESD	D_{mis}	$S_{\overline{x}}$	CART	LR	SVM		REM	RESD	D_{mis}	$S_{\overline{x}}$	CART	LR	SVM	
Deletion	0.01	0.03	-	-	0.69	0.68	0.68	2/7	0.05	0.03	-	-	0.75	0.73	0.74	2/7
Imputation	0.06	0.37	-	-	1	0.89	0.89	0/7	0.05	0.37	-	-	0.68	0.75	0.74	1/7
MisGAN [23]	1.56	0.27	0.77	0.97	0	0	0	0/7	1.06	0.98	2	h	0	0	0	0/7
BN [46]	31.53	16.53	0.50	1.48	1	0.85	0.85	0/7	22.49	452.52	0.64	h	0.90	0.81	0.75	1/7
CTGAN [31]	238.98	106.07	0.09	1.58	1.09	0.93	0.93	1/7	25.51	103.74	0.78	h	0.75	0.59	0.65	0/7
fRand	0.01	0.04	0.01	0.38	0.69	0.68	0.68	3/7	0.05	0.08	2	h	0.55	0.77	0.73	1/7
pRand	0.01	0.03	0	0.37	0.69	0.68	0.68	3/7	0.05	0.08	0	0.56	0.75	0.74	0.75	2/7
MergeGAN	0.14	0.09	0.06	0.32	1	0.97	0.97	0/7	0.05	0.08	0.37	0.34	0.68	0.75	0.74	2/7
MergeVAE	0.13	0.55	0	0.26	0.69	0.68	0.68	1/7	0.01	0.06	0	0.17	0.52	0.54	0.53	4/7
MergeBN	0.13	0.45	0	0.07	1	0.99	0.97	3/7	0.01	0.06	0	0.14	0.89	0.92	0.93	7/7
HottGAN	0.02	0.11	0	0.28	1.04	0.99	0.99	5/7	0.05	0.08	0	0.27	0.77	0.85	0.85	2/7
HottVAE	0.13	0.34	0	0.09	0.98	0.92	0.92	1/7	0.01	0.13	0	0.21	0.52	0.54	0.53	2/7
HottBN	0.03	0.10	0	0.05	1	0.98	0.98	4/7	0.09	223.91	0	0.68	0.89	0.91	0.91	4/7

Since MisGAN only produced one value of the target binary variable, the AUROC is undefined; so we use 0 to depict its worst performance.
h depicts a high value of $S_{\overline{x}}$ when the number of MPs generated are way more that that of in the real data.

relative to that of real data, i.e., AUROC for Training over Real and Testing over Real (TRTR). In Table II, we report proportional AUROC, denoted as pAUROC and computed as $(\text{AUROC[TRTS]} + \text{AUROC[TSTR]})/(2 * \text{AUROC[TRTR]})$. The higher the *pAUROC*, the higher the quality of synthetic data. Each classifier is trained over 80% of data and tested on the rest 20%, five times with random splits.

Results. Let us begin with the t-SNE plots (Fig. 5). Since the Scikit-learn t-SNE plotting implementation could not handle large datasets, we used samples of size $5,000$ for each missing pattern for real and synthetic datasets. We picked the samples from real and synthetic that were closest to each other. For Price, we see that all methods are able to mimic the real data distribution for the complete samples (i.e., MP4) except for MisGAN. MisGAN generates extremely bad quality data for all MPs—and it is the worst performing method on all evaluation metrics (see Table II).

Although Deletion and Imputation are able to do well for MP4, they cannot generate data for other missing patterns (as discussed earlier). Furthermore, our *mp*-distribution preserving methods outperform them, giving lower (or comparable) errors and much higher pAUROC scores for classification (Table II). For instance, pRand has the performance as Deletion, but additionally preserves *mp*-distribution (see D_{miss} and $S_{\overline{x}}$); and HottGAN and HottBN outperforms Imputation on every single evaluation criterion.

Further looking at the t-SNE plots reveals that Bayesian Network performs better than MisGAN as well as pRand and fRand. It however commits the highest errors for mean and standard deviation (see REM and RES in Table II), which suggests replacing the missing data with a special value distorts the data distribution significantly. Nevertheless, for the downstream task, BN outperforms Deletion, fRand, pRand, and MisGAN, all of which rely on complete samples and MCAR missingness. But, here again (i.e., for the downstream task) HottBN and MergeBN both outperform BN, showing a clear advantage of preserving *mp*-distribution.

Going from Price to Brain dataset, fRand and MisGAN are quite useless: both produce lots of MPs—MisGAN and fRand respectively produced 889 and 106681 MPs—but none of them were the actual MPs present in the real data. This is because when missingness occurs in more features, the methods that

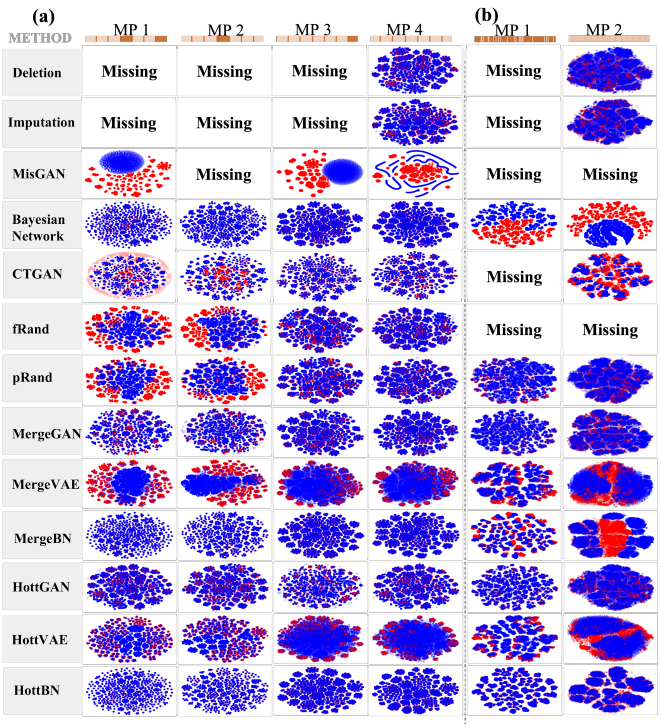

Fig. 5: t-SNE plot, Price (a) and Brain (b) datasets. Each plot gives a "scatter-plot" projection of original data (red points) and synthetic data (blue points) for each missing pattern (MP). The higher the overlap between the red and blue points, the higher the synthetic data quality.

assume MCAR missingness produce more missing patterns. Thus, they are not appropriate for non-MCAR settings. Although BN was able to generate data for the two MPs in Brain data, it resulted in higher D_{mis} (Table II): it produced 1334 MPs, with 73% fewer records for MP1 and 52% more records for MP2. Furthermore, for the downstream task, BN performs better than all other methods except for HottGAN, HottBN, and MergeBN. Yet in terms of other statistical data quality measures, BN is still among the worst methods.

Across both datasets, HottBN and MergeBN have the best overall performance as indicated by the scores and individual evaluation criteria, while HottGAN remains the second best option. When the missing patterns are too many to have

sufficient support for individual hott partitions, MergeGAN is expected to perform better than HottGAN.

We further evaluated HottGEN+ on two real-world datasets, Price and Brain, using all the missing patterns (8 for Price and 53 for Brain). In addition to the datasets generated by HottGAN, we used MergeGAN to generate synthetic datasets for the remaining patterns. Specifically, we calculated the D_{mis} and $S_{\overline{x}}$ metrics for HottGAN+MergeGAN. The results show that HottGEN+ outperforms HottGAN in both cases, with improvements of 9.8% and 21.4% on $S_{\overline{x}}$ for Price (0.174 vs. 0.193) and Brain (0.198 vs. 0.252), respectively. Additionally, there were improvements on D_{mis} for both datasets (Price: 0 vs. 0.002; Brain: 0 vs. 0.0106). Note that the performance of HottGAN reported here is different from the values reported in Table II , as the datasets in Table II excluded the minor missing patterns.

Remarks: The use of BN gives an interesting case underscoring the importance of preserving the *mp*-distribution in synthetic data generation. Our vanilla version of BN adapts traditional Bayesian Networks to handle missing data internally, eliminating the need for imputation or deletion of samples with missing values. However, this standard version underperforms compared to our enhanced versions corresponding to Hott Partitioning (HottBN) and Merge Approach (MergeBN).

For example, in Table II, the vanilla version of BN scores 1/7 for both the Price and Brain datasets. In contrast, our enhanced versions significantly improve performance, achieving scores of 4/7 to 7/7. This improvement is achieved by preserving both observable and missing data distributions. The impact and innovation of our methods are evident in the substantial performance gains illustrated in Table I, Figure C.1a, Figure C.1b, and Table II. Furthermore, the suite of approaches presented in this work can handle a variety of missing data scenarios, induced by different types of missing data mechanisms and the resulting *mp*-distributions. In particular, the HottGEN+ approach, which uses the hott partitioning on the top-k missing patterns, combined with other approaches like MergeGEN for the remaining data, provides an effective method to calibrate for a *mp*-distribution. This boosts the utility of the generated synthetic data (Figure 4).

VII. Discussion and Conclusion

Real data often exhibits missing values, which have traditionally been deemed problematic and removed via deletion or imputation methods, under the assumption that a dataset without missing values (after imputation) could enhance the performance of machine learning models. This is often justified when the missing mechanism is MCAR, but not otherwise. Many online tutorials for machine learning and data analysis assert that missing data imputation is essential in the standard data processing pipeline, with simple methods such as mean imputation and kNN being the most widely adopted methods.

However, treating all missing data merely as noise is fundamentally flawed. Handling missing data is inherently complex: simple imputation methods typically fail to address complex scenarios such as MNAR while most evaluations of imputation methods assume that all missing values are from the same missing-data mechanism. Crucially, without careful consideration, imputed datasets might introduce or even amplify biases. Synthetic datasets, designed to mimic real datasets, inevitably inherit and propagate these biases, potentially leading to unforeseen consequences, particularly when used to train future generative models on Web-crawled datasets. In this work, we introduce three novel approaches, HottGEN, MergeGEN, and HottGEN+, that retain the distributions of both observable and missing data, thereby creating more realistic datasets and enhancing the flexibility of synthetic data applications. For instance, when such data is used to train machine learning models, the most suitable imputation method can be employed to minimize bias. Moreover, our approach specifically addresses the problem of missing data, particularly under non-random missing mechanisms, which existing methods often fail to properly handle. By explicitly modeling and replicating missing patterns, we ensure that the missing data distribution is faithfully preserved in the synthetic datasets, as opposed to other baseline methods which can mirror the observable data distribution but do not explicitly account for the structure or patterns of missing data, resulting in a loss of important information.

Furthermore, as demonstrated before and in a recent study [7], real data often displays distinct missing patterns that are not random. These patterns, although informative, are often ignored by existing generative models. In this paper, we introduce methods that efficiently leverage these missing patterns and generate synthetic data that preserves both the missing pattern distribution as well as the observable data distribution of the real data. Our empirical evaluation shows significant improvements in synthetic data quality, for instance, HottBN achieved 96% improvements in terms of $S_{\overline{x}}$ on Price dataset, compared to the traditional simple imputation method.

In the future, we plan to tackle temporal data generation, which poses unique challenges due to the temporal correlations in missing data distributions. Also, given our method's capacity to learn data distributions conditioned on missing patterns, we aim to explore its potential in data imputation.

References

[1] Z. Wang, P. Myles, and A. Tucker, "Generating and evaluating synthetic uk primary care data: preserving data utility & patient privacy," in *2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)*. IEEE, 2019, pp. 126–131.

[2] J. Vaidya, B. Shafiq, M. Asani, N. Adam, X. Jiang, and L. Ohno-Machado, "A scalable privacy-preserving data generation methodology for exploratory analysis," in *AMIA Annual Symposium Proceedings*, vol. 2017. American Medical Informatics Association, 2017, p. 1695.

[3] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, "Data synthesis based on generative adversarial networks," *Proceedings of the VLDB Endowment*, 2018.

[4] F. Ventura, Z. Kaoudi, J. A. Quiané-Ruiz, and V. Markl, "Expand your training limits! generating training data for ml-based data management," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1865–1878.

[5] C. Ge, S. Mohapatra, X. He, and I. F. Ilyas, "Kamino: Constraint-aware differentially private data synthesis," *Proc. VLDB Endow.*, vol. 14, no. 10, p. 1886–1899, jun 2021. [Online]. Available: https://doi.org/10.14778/3467861.3467876

[6] M. Mangel and F. J. Samaniego, "Abraham wald's work on aircraft survivability," *Journal of the American Statistical Association*, vol. 79, no. 386, pp. 259–267, 1984.

[7] R. Mitra, S. F. McGough, T. Chakraborti, C. Holmes, R. Copping, N. Hagenbuch, S. Biedermann, J. Noonan, B. Lehmann, A. Shenvi *et al.*, "Learning from data with structured missingness," *Nature Machine Intelligence*, vol. 5, no. 1, pp. 13–23, 2023.

[8] C. I. Braem, U. S. Yavuz, H. J. Hermens, and P. H. Veltink, "Missing data statistics provide causal insights into data loss in diabetes health monitoring by wearable sensors," *Sensors*, vol. 24, no. 5, p. 1526, 2024.

[9] A. Lupattelli, M. E. Wood, and H. Nordeng, "Analyzing missing data in perinatal pharmacoepidemiology research: methodological considerations to limit the risk of bias," *Clinical Therapeutics*, vol. 41, no. 12, pp. 2477–2487, 2019.

[10] H. Asif and J. Vaidya, "A study of users' privacy preferences for data sharing on symptoms-tracking/health app," in *Proceedings of the 21th Workshop on Privacy in the Electronic Society*. ACM, 2022.

[11] Y. Zhang, Z. Gan, and L. Carin, "Generating text via adversarial training," in *NIPS workshop on Adversarial Training*, vol. 21. academia. edu, 2016, pp. 21–32.

[12] P. E. McKnight, K. M. McKnight, S. Sidani, and A. J. Figueredo, *Missing data: A gentle introduction*. Guilford Press, 2007.

[13] C. Ma and C. Zhang, "Identifiable generative models for missing not at random data imputation," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[14] D. T. Neves, J. Alves, M. G. Naik, A. J. Proença, and F. Prasser, "From missing data imputation to data generation," *Journal of Computational Science*, p. 101640, 2022.

[15] Y. Yang, Z. Xu, and D. Song, "Missing value imputation for microrna expression data by using a go-based similarity measure," in *BMC bioinformatics*, vol. 17, no. 1. BioMed Central, 2016, pp. 109–116.

[16] K. Moorthy, A. N. Jaber, M. A. Ismail, F. Ernawan, M. S. Mohamad, and S. Deris, "Missing-values imputation algorithms for microarray gene expression data," *Microarray Bioinformatics*, pp. 255–266, 2019.

[17] Y. Zhang, Y. Wang, and S. Wang, "Improvement of collaborative filtering recommendation algorithm based on intuitionistic fuzzy reasoning under missing data," *IEEE Access*, vol. 8, pp. 51 324–51 332, 2020.

[18] W. Kweon and H. Yu, "Doubly calibrated estimator for recommendation on data missing not at random," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 3810–3820.

[19] J.-H. Lin and P. J. Haug, "Exploiting missing clinical data in bayesian network modeling for predicting medical problems," *Journal of biomedical informatics*, vol. 41, no. 1, pp. 1–14, 2008.

[20] J. T. Chi, E. C. Chi, and R. G. Baraniuk, "k-pod: A method for k-means clustering of missing data," *The American Statistician*, vol. 70, no. 1, pp. 91–99, 2016.

[21] K. Wagstaff, "Clustering with missing values: No imputation required," in *Classification, clustering, and data mining applications*. Springer, 2004, pp. 649–658.

[22] P. J. García-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal, "Pattern classification with missing data: a review," *Neural Computing and Applications*, vol. 19, no. 2, pp. 263–282, 2010.

[23] S. C.-X. Li, B. Jiang, and B. Marlin, "Misgan: Learning from incomplete data with generative adversarial networks," *arXiv preprint arXiv:1902.09599*, 2019.

[24] A. Tucker, Z. Wang, Y. Rotalinti, and P. Myles, "Generating high-fidelity synthetic patient data for assessing machine learning healthcare software," *NPJ digital medicine*, vol. 3, no. 1, pp. 1–13, 2020.

[25] X. Wang, H. Asif, and J. Vaidya, "Preserving missing data distribution in synthetic data," in *Proceedings of the ACM Web Conference 2023*, ser. WWW '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 2110–2121. [Online]. Available: https://doi.org/10.1145/3543507.3583297

[26] G. Forestier, F. Petitjean, H. A. Dau, G. I. Webb, and E. Keogh, "Generating synthetic time series to augment sparse datasets," in *2017 IEEE international conference on data mining (ICDM)*. IEEE, 2017, pp. 865–870.

[27] J. Dahmen and D. Cook, "Synsys: A synthetic data generation system for healthcare applications," *Sensors*, vol. 19, no. 5, p. 1181, 2019.

[28] C. Ge, S. Mohapatra, X. He, and I. F. Ilyas, "Kamino: Constraint-aware differentially private data synthesis," *Proceedings of the VLDB Endowment*, vol. 14, no. 10, pp. 1886–1899, 2021.

[29] K. Cai, X. Lei, J. Wei, and X. Xiao, "Data synthesis via differentially private markov random fields," *Proceedings of the VLDB Endowment*, vol. 14, no. 11, pp. 2190–2202, 2021.

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

[31] L. Xu and K. Veeramachaneni, "Synthesizing tabular data using generative adversarial networks," *arXiv preprint arXiv:1811.11264*, 2018.

[32] W. Li, "Supporting database constraints in synthetic data generation based on generative adversarial networks," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 2875–2877.

[33] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[34] J. Yoon, J. Jordon, and M. Schaar, "Gain: Missing data imputation using generative adversarial nets," in *International conference on machine learning*. PMLR, 2018, pp. 5689–5698.

[35] H. Li, Y. Liao, Z. Tian, Z. Liu, J. Liu, and X. Liu, "Bidirectional stackable recurrent generative adversarial imputation network for specific emitter missing data imputation," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 2967–2980, 2024.

[36] H. Chen, Y. Jiang, S. Guo, X. Mao, Y. Lin, and H. Wan, "Difflight: A partial rewards conditioned diffusion model for traffic signal control with missing data," *Advances in Neural Information Processing Systems*, vol. 37, pp. 123 353–123 378, 2025.

[37] D. B. Rubin, *Multiple imputation for nonresponse in surveys*. John Wiley & Sons, 2004, vol. 81.

[38] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[39] T. Kokosi, B. De Stavola, R. Mitra, L. Frayling, A. Doherty, I. Dove, P. Sonnenberg, and K. Harron, "An overview on synthetic administrative data for research," *International Journal of Population Data Science*, vol. 7, no. 1, 2022.

[40] A. Yale, S. Dash, R. Dutta, I. Guyon, A. Pavao, and K. Bennett, "Privacy preserving synthetic health data," in *ESANN 2019-European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2019.

[41] Z. Lin, V. Sekar, and G. Fanti, "On the privacy properties of gan-generated samples," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 1522–1530.

[42] I. Deshpande, Z. Zhang, and A. G. Schwing, "Generative modeling using the sliced wasserstein distance," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3483–3491.

[43] S. Kolouri, K. Nadjahi, U. Simsekli, R. Badeau, and G. Rohde, "Generalized sliced wasserstein distances," in *Advances in Neural Information Processing Systems*, 2019, pp. 261–272.

[44] J. Wang, R. Gao, and Y. Xie, "Two-sample test using projected wasserstein distance," in *2021 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2021, pp. 3320–3325.

[45] S. Prakash, "Brain tumor dataset," https://www.kaggle.com/datasets/sooryaprakash12/texephyr, 2021.

[46] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[47] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[48] M. Scutari, "Bayesian network constraint-based structure learning algorithms: Parallel and optimized implementations in the bnlearn R package," *Journal of Statistical Software*, vol. 77, no. 2, pp. 1–20, 2017.

[49] A. Goncalves, P. Ray, B. Soper, J. Stevens, L. Coyle, and A. P. Sales, "Generation and evaluation of synthetic patient data," *BMC medical research methodology*, vol. 20, no. 1, pp. 1–40, 2020.

[50] R. J. Little, "A test of missing completely at random for multivariate data with missing values," *Journal of the American statistical Association*, vol. 83, no. 404, pp. 1198–1202, 1988.

[51] N. Tierney, D. Cook, M. McBain, and C. Fay, *naniar: Data Structures, Summaries, and Visualisations for Missing Data*, 2021, r package version 0.6.1. [Online]. Available: https://CRAN.R-project.org/package=naniar

[52] K. Singh, "Retail prices of commodities in india," https://www.kaggle.com/datasets/kk9969/retail-prices-of-commodities-in-india, 2021.

[53] R. Timofeev, "Classification and regression trees (cart) theory and applications," *Humboldt University, Berlin*, vol. 54, 2004.

[54] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[55] N. Gao, H. Xue, W. Shao, S. Zhao, K. K. Qin, A. Prabowo, M. S. Rahaman, and F. D. Salim, "Generative adversarial networks for spatio-temporal data: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 2, pp. 1–25, 2022.

[56] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[57] Y. Bai, T. Ma, and A. Risteski, "Approximability of discriminators implies diversity in gans," *arXiv preprint arXiv:1806.10586*, 2018.