DDGANI: Missing Value Imputation using Denoising Diffusion GAN with Modular Plugins (Technical Report)

Shuang Hao, Xiang Huang
School of Computer Science and Technology, Beijing Jiaotong University, China
Beijing Key Laboratory of Traffic Data Analysis and Mining, China
{haoshuang,22120382}@bjtu.edu.cn

ABSTRACT

Dealing with missing values in relational tables is a pervasive challenging preparation phase before proceeding to data analysis and mining. Directly removing missing data often causes the loss of information and thus adversely impacts further data processing. Missing values can also be estimated using extensively studied imputation techniques. However, they seldom consider improving the utility of imputed data for the downstream target task. Recently widely used learning-based approaches rarely try to maintain the inherent data dependencies. In this paper, we introduce DDGANI, a novel missing value imputation framework based on denoising diffusion GAN. Central to our approach is the utilization of denoising diffusion GAN for effective data imputation. Additionally, we incorporate a self-attention mechanism, enabling the model to focus on more relevant information from similar tuples. Our framework also integrates two plugins that can be chosen based on data characteristics and downstream tasks, which ensure the utility of imputed data and the preservation of inherent data dependencies. Extensive experiments on multiple datasets demonstrate the superiority of DDGANI compared to existing state-of-the-art imputation techniques and verify the effectiveness of two modular plugins.

PVLDB Reference Format:
Shuang Hao, Xiang Huang. DDGANI: Missing Value Imputation using Denoising Diffusion GAN with Modular Plugins (Technical Report). PVLDB, 14(1): XXX-XXX, 2024.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/xianghuang2/DDGANI.

1 INTRODUCTION

Data is undoubtedly an essential element of decision-making, so high-quality data is a necessity. However, in real-world scenarios, data often goes missing, which may arise due to human error during data collection or equipment malfunctions. While some factors leading to missing values are perceived, they are unavoidable, such as table joining for relational data augmentation [11]. Therefore, it is essential to address the missing data before diving into data analysis and mining.

Existing Approaches and Limitations. Manually imputing these missing values is undoubtedly a time-consuming and labor-intensive task. A straightforward approach might be to delete these missing data. However, neglecting such missing values can distort the analysis or lead to inaccuracies, ultimately affecting the downstream decision-making process [3]. Thus, the most common strategy to address missing values is reconstructing a complete dataset through data imputation techniques. Most of the earlier studies focus on the basic statistical imputation methods, utilizing statistics such as the mean, median, and mode to impute the missing values. However, most of these methods only work well for numerical data. The traditional machine learning (ML) imputation solutions train an ML model for missing value prediction. The accuracy of imputation is limited by the predictive power of the ML model. In addition, some of them have strict distributional assumptions such as PCAI [23], and some are less effective for categorical data such as KNN [1]. Heuristic-based methods utilize data imputation rules to deal with missing values, such as the relaxed functional dependencies [7] and differential dependencies [41]. However, they mainly focus on repairing categorical values (*e.g.,* City, State), and the design of high-quality imputation rules requires extensive domain knowledge [29]. Moreover, they sometimes fail to provide a deterministic imputation solution for attributes on the left-hand side of these data dependencies. For example, for the missing value t_5[AreaCode] in Table 1, these methods may give two possible imputations as both "501" and "870" satisfy the given imputation rules.

Recently, deep learning based approaches, mainly consisting of methods based on deep generative models and pre-trained language models, have provided an effective way to tackle the above obstacles and have obtained impressive results on the missing value imputation problem. Generative adversarial networks (GAN) and variational autoencoder (VAE) are two common deep generative models adopted in previous work [18, 31, 35, 51]. However, they have difficulty directly generating samples from a complex distribution in one shot. Furthermore, they often neglect to check whether the imputed table violates the original data dependencies. For example, given the similar distribution between t_4 and t_5 in Table 1, most of these methods would impute t_4[State] as "AR" which results in a violation of φ_1. Actually, "Neward" in t_4 and t_5 are different cities in different states despite having the same name. With the prominence of pre-trained language models (PLMs) on multiple data processing tasks, there are PLMs-based data imputation methods such as TURL [16] and RPT [45]. TURL primarily focuses on learning data representations and aligning these representations with knowledge bases to retrieve missing data. RPT adopts a Transformer-based

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

Table 1: Example of a Relational Table with Missing Values
(A gray cell means it is missing and an turquoise colored word indicates the true value.)
($\varphi_1 : Areacode \rightarrow State$, $\varphi_2 : Areacode, MaritalStatus \rightarrow SingleExemp$)

Tuple ID	Fname	AreaCode	City	State	MaritalStatus	SingleExemp	Salary	Rate	HasChild (label)
t_1	Muneo	201	Hampton	NJ	M	0	10000	1.40	Yes
t_2	Hyujin	201	Hampton	NJ	M	0	[90000]	2.50	Yes
t_3	Jarno	201	Hampton	NJ	S	[1000]	80000	2.34	No
t_4	Coke	201	Newark	[NJ]	M	0	5000	[1.40]	No
t_5	Ada	[501]	Newark	AR	M	0	5000	1.38	Yes
t_6	Liesbeth	501	Newark	AR	S	20	[5000]	1.38	No
t_7	Refik	501	[Newark]	AR	S	20	10000	2.58	Yes
t_8	Sreerupa	870	Sage	AR	[S]	20	[5000]	1.38	No

neural translation architecture, pre-trained by corrupting the input tuple and then learning to reconstruct the original tuple to support the missing value imputation task. However, PLMs-based imputation methods do not handle numerical data well (*e.g.,* Rate). Furthermore, they often fail to ensure the applicability of the imputed data for subsequent tasks, which is a significant challenge at present [34].

Our Proposal. To address the aforementioned limitations, we propose a novel missing value imputation framework DDGANI as depicted in Figure 1. It contains a data imputation module, which is the mainstay of the framework, and two optional plug-ins: a data utility plug-in and a data dependency plug-in. We employ the denoising diffusion GAN [50] in the data imputation module to reconstruct the complete dataset, which takes the relational table with missing values as input and outputs the imputed table. This choice is motivated by its ability to model complex data distributions through several conditional denoising diffusion steps, allowing for the proficient generation of various types of data and achieving high data quality while also obtaining good mode coverage. Additionally, we incorporate the self-attention mechanism, which enables the generative model to capture more relevant contextual information when dealing with missing data.

We introduce a data utility plug-in to ensure the utility of the imputed data for the subsequent classification task via a meta-learning paradigm. We fully use the downstream classifier and validation set, with the goal that the classifier trained with imputed data could minimize the loss on the validation set. It is achieved by backpropagating the validation loss to the data generation module to guarantee the utility of the imputed table. If the downstream classifier or validation set is unavailable, the data utility plug-in builds a two-layer fully connected neural network acting as the downstream classifier and selects some tuples from the incomplete table where all attributes are observed to serve as a validation set. This can additionally contribute to keeping the same data distribution between the imputed and observed data.

We also design a data dependency plug-in to allow the imputed data to adhere to the original data dependencies among the categorical attributes. In this paper, we describe how it works using function dependencies (FDs) as an example. This plug-in could also support other constraints such as conditional functional dependencies (CFDs). This module initially mines potential FDs from the table with missing values. Each FD $\varphi : Z \rightarrow A$ is then modeled by a classifier, which takes the values of attributes Z in the left-hand side of the FD as input and predicts the attribute value of A (These models also could be pre-trained using the master data or knowledge bases). We minimize the difference between the predicted values given by the dependency models and the imputed values given by the data imputation module to keep the imputed data from violating the existing data constraints. For example, given φ_1 : AreaCode→State, the imputation of t_4[State] in Table 1 with "AR" would result in a penalty in the loss of the generative model. Consequently, DDGANI would lean towards imputing it with "NJ". However, the automatically generated FDs from an incomplete dataset may not be fully trusted. As the imputation becomes more accurate, we must go back to determine whether these FDs are valid based on the confidence of constraints and violated tuples. Intuitively, if the violated tuples are more trusted, we refine the FDs and their corresponding dependency models.

Each component in our framework, though appearing to have its distinct goal, works together to minimize imputation error. The attention module and two plug-ins further improve the ability of the base model to capture the correlations between attributes and between attributes and the label in different ways. Meanwhile, this modular design can make the whole framework more flexible for different data characteristics and downstream tasks.

Contributions. Our contributions are summarized as follows:

- We propose a novel missing value imputation framework with a self-attention-enhanced generative model for tabular data imputation and two plug-ins that can be chosen to improve the performance based on data characteristics and downstream tasks.
- We adopt the denoising diffusion GAN for missing value imputation. To the best of our knowledge, we are the first effort to model the complex data distribution implied in missing data through denoising diffusion steps to obtain high-quality imputation results (Section 4).
- We design a data utility plug-in to boost the performance of our imputation framework via a meta-learning paradigm, which guarantees the applicability of our imputed data for training downstream models for a target task (Section 5).
- We propose a data dependency plug-in to ensure the preservation of inherent data dependencies in the imputed relational table. We iteratively impute the missing values and refine the dependency models based on the confidence of mined FDs and imputed tuples. (Section 6).

- We conducted extensive experiments on multiple datasets with both categorical and numerical attributes. The experimental results demonstrate that our approach outperforms a variety of state-of-the-art missing value imputation methods. (Section 8).

2 RELATED WORK

We categorize related work as follows. Note that this work extends our short paper [19] by including 1) a new choice on the missing value imputation model, *i.e.,* we employ the denoising diffusion GAN instead of VAE, which stems from the fact that the multi-step diffusion process tends to generate data of higher quality; 2) a more comprehensive attention module, which takes into account the correlations between different attributes instead of just at the tuple level; 3) a data dependency plug-in to ensure the preservation of inherent data dependencies in the relational table; 4) more comprehensive empirical evaluations, mainly including more datasets and baselines, the effectiveness of newly added plug-ins, the impact of hyper-parameters, the capability to handle rare values, and the runtime analysis.

Statistical Imputation Approaches. Statistical-based methods typically calculate statistics such as mean, median, and mode of observed data to estimate missing values. Several statistical methods can also be combined to impute missing data [8, 48]. However, most of these methods only work well for imputing numerical data, which is unsuitable for some real-world scenarios and far from enough to be high-quality.

Heuristic Based Imputation Approaches. Heuristic methods have been proposed to effectively estimate the missing values without violating the integrity of the original data. For example, Derand [41], RENUVER [7], Graf [36] and Holistic [14] respectively utilize differential dependencies [40], relaxed functional dependencies [10], conditional functional dependencies [6] and denial constraints [13] to ensure the imputed data satisfy the given constraints. However, they cannot impute the attributes not appearing in the dependencies. There exist some methods directly mining constraints from missing data for the subsequent imputation [4, 27] and estimating the confidence (*i.e.,* genuineness score) of discovered constraints based on the tuple frequency while ignoring each tuple's confidence. Graf [36] iteratively repairs the dirty data and inaccurate CFDs, but it requires lots of complete tuples for constraint learning.

Machine Learning Based Imputation Approaches. ML-based imputation methods generally train ML models to predict the missing values. The k-nearest neighbors (KNN) based method [47] uses a weighted average of values from the k nearest neighbors to estimate the missing value. MissForest [43] predict the missing values utilizing observed data through the random forest. Other ML-based methods exist, such as MICE [38] and PCAI [30]. Although these ML-based imputation methods outperform statistical ones [33], they may have strict distributional assumptions or struggle with categorical data.

Deep Learning Based Imputation Approaches. DL-based methods generally outperform the aforementioned approaches as they can capture the correlations between data through multiple layers of non-linear computations, enabling effective imputation of missing data. DataWig [5] builds a DL model for each attribute to predict the missing values. GAIN [51] and RandomGAN [18] adapt GAN architecture. HI-VAE [35] and VAEI [32] extend VAE to impute missing data. GINN [42] and EEG-GAE [46] utilize graph neural networks. Pre-trained language models have recently been used for data imputation tasks, including TURL [16] and RPT [45]. Baran [28] also employs pre-trained DL models. Meanwhile, it iteratively asks the user to label the data and leverages the provided user corrections to update the pre-trained models incrementally. Some studies such as AimNet [49] impute the missing values based on DL models while incorporating data dependencies. Our work falls into the scope of DL-based methods, and the experimental results show that our method outperforms the above baselines.

3 PROBLEM FORMULATION

Data Model. Let $\boldsymbol{X}$ denote a relational table with n tuples $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and k attributes $U = \{A_1, \ldots, A_k\}$. Each attribute is either numerical or categorical. We use x_{ij} to denote the value of attribute A_j in tuple $\boldsymbol{x}_i$. Each tuple $\boldsymbol{x}_i$ has a label y_i for classification.

Note that for ease of illustration in the examples, we sometimes refer to each tuple by its ID t_i in Table 1 and use $t_i[A_j]$ to denote the value of attribute A_j in tuple t_i.

Missing Data. $\boldsymbol{X}$ contains *missing data*: each tuple $\boldsymbol{x}$ may have some attribute values that are not observed. To formalize these missing values, we follow the existing work [26] to introduce a *mask* matrix $\boldsymbol{M} = \{\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n\}$ to indicate which value in $\boldsymbol{X}$ is observed. Each $\boldsymbol{m}_i = \{m_{i1}, \ldots, m_{ik}\}$ represents the missing status of tuple $\boldsymbol{x}_i$: x_{ij} is observed if $m_{ij} = 1$, otherwise x_{ij} is missing.

Example 3.1. Given a relational table in Table 1 where a gray cell means it is missing. For tuple t_4 in Table 1, it has missing values in t_4[State] and t_4[Rate] while other attribute values are observed. Therefore, its corresponding mask is $\boldsymbol{m}_4 = \{1, 1, 1, 0, 1, 1, 1, 0\}$.

Data Dependency. A dependency is a constraint that applies to or defines the relationship between attributes, where knowing the value of one attribute (or a set of attributes) is enough to infer the value or range of another attribute. Let's take the functional dependency (FD) as an example. We say $\boldsymbol{X}$ satisfies the FD $\varphi : Z \rightarrow A$ (Z, A are two disjoint subsets of the attributes, and the right-hand side A of φ has only one attribute) if, for any tuple, the attribute values in Z uniquely determine the attribute value in A.

Example 3.2. Consider the following two functional dependencies in Table 1:

- φ_1 : AreaCode$\rightarrow$State
- φ_2 : State, MaritalStatus$\rightarrow$SingleExemp

where φ_1 means that AreaCode uniquely determines State and there should not exist two tuples that have the same value in AreaCode but different values in State.

Missing Data Imputation. The missing data imputation problem is to replace missing data in $\boldsymbol{X}$ with an estimated value based on other observed values. If $\bar{\boldsymbol{X}} = \{\bar{x}_1, \ldots, \bar{x}_n\}$ denotes the imputed table and $\boldsymbol{X}^{\checkmark} = \{x_1^{\checkmark}, \ldots, x_n^{\checkmark}\}$ denotes the complete ground-truth table, which is not available in practical situations, our goal is to minimize the difference between $\bar{\boldsymbol{X}}$ and $\boldsymbol{X}^{\checkmark}$.

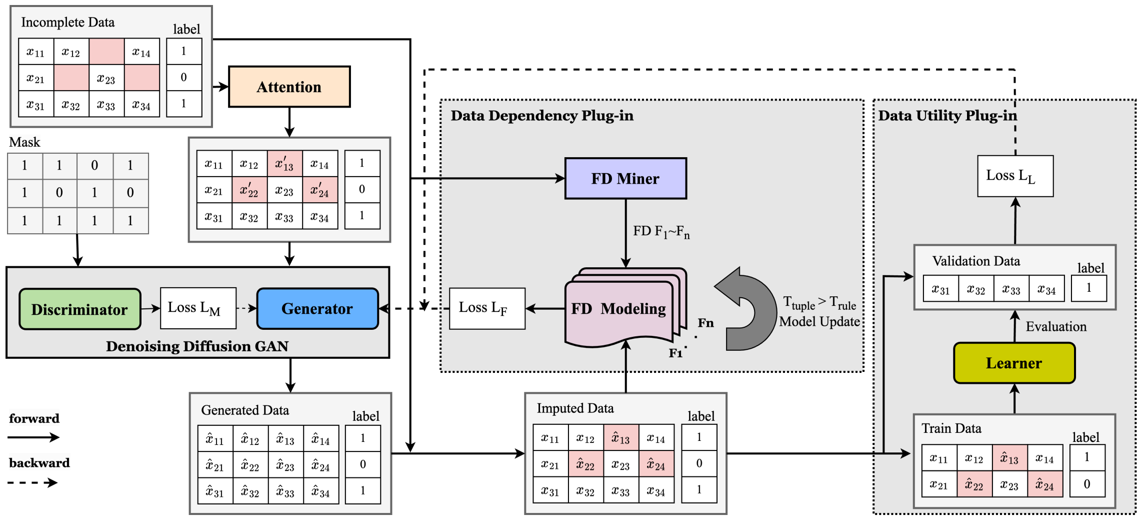


Figure 1: The Architecture of DDGANI

4 DENOISING DIFFUSION GAN BASED IMPUTATION NETWORK

In this section, we will start with an introduction to denoising diffusion GAN and then discuss how to utilize it to impute the missing values in a relational table.

4.1 Denoising Diffusion GAN

Why Diffusion Models? Diffusion models have been shown to outperform generative models in a one-shot manner in many cases, despite the complexity of their training and convergence as a trade-off. Kotelnikov et al. [25] have demonstrated that the diffusion models could better learn the distribution characteristics of tabular data. This is because the one-shot generative processes such as GANs and VAEs face the difficulty of directly generating samples from a complex distribution in one shot, which might not capture all data details. In contrast, diffusion models first establish a coarse distribution and then incrementally refine it, ensuring high-quality data generation while obtaining good mode coverage.

Denoising Diffusion Probabilistic Model. The denoising diffusion probabilistic model (DDPM) defines a forward diffusion process that adds varying degrees of Gaussian noise to the original data $\boldsymbol{X}_0 \sim q(\boldsymbol{X}_0)$ over T steps where $q(\boldsymbol{X}_0)$ is the original data distribution. The forward process at step t can be represented as $q(\boldsymbol{X}_t|\boldsymbol{X}_{t-1})$ where $\boldsymbol{X}_t$ denotes the data obtained by adding noise to $\boldsymbol{X}_{t-1}$, *i.e.,* to $\boldsymbol{X}_0$ for t steps. The reverse diffusion process is to guess the Gaussian noise added in the forward process and remove it iteratively (within T steps), using a neural network model Φ_θ. The reverse process can be represented as $p_\theta(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t)$ where θ represents the parameters of the model. Thus, the training objective of the diffusion-based generative model amounts to maximizing the log-likelihood of the data generated at the end of the reverse process belonging to the original data distribution, which is formulated as maximizing the evidence lower bound (ELBO) [24].

However, due to the typically unknown nature of $q(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t)$ in the diffusion model, the denoising step size T is assumed to be on the order of thousands of steps, and the diffusion rate β_t in each step should be small, which makes its training time-consuming.

Denoising Diffusion GAN. The large denoising diffusion steps T leads to significant computational costs during model training. To address this issue, Xiao et al. [50] employed conditional GANs to approximate $q(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t)$. Specifically, the generator G takes a random noise as input and outputs $\hat{\boldsymbol{X}}_0$ with $\boldsymbol{X}_t$ and step t as conditions. Then, $\hat{\boldsymbol{X}}_{t-1}$ is obtained through a posterior sampling process $q(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t, \boldsymbol{X}_0)$ from $\hat{\boldsymbol{X}}_0$. The discriminator D takes $(\hat{\boldsymbol{X}}_{t-1}, \boldsymbol{X}_t, t)$ or $(\boldsymbol{X}_{t-1}, \boldsymbol{X}_t, t)$ as inputs and determines the probability that the input data is $\boldsymbol{X}_{t-1}$ (real) rather than $\hat{\boldsymbol{X}}_{t-1}$ (fake) given $\boldsymbol{X}_t$. After achieving stability in training, the generated $\hat{\boldsymbol{X}}_{t-1}$ from the generator will closely resemble $\boldsymbol{X}_{t-1}$, resulting in a consistent distribution between $p_\theta(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t)$ and $q(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t)$. Therefore, denoising diffusion GANs could adopt a relatively large diffusion rate β_t and a small denoising diffusion step T ($T \leq 8$).

4.2 Denoising Diffusion GAN For Data Imputation

Now it's time to introduce how to impute the missing values with denoising diffusion GAN.

Attention Mechanisms. Instead of feeding the data $\boldsymbol{X}$ with missing values directly into the generative model, we pre-process $\boldsymbol{X}$ with the self-attention mechanism, which encourages the subsequent generation model to capture the internal dependencies within the data and pay more attention to the observed data associated with the missing values.

Here, we build an attention map $\mathcal{A}^j$ for each attribute A_j, of size $n \times n$ where n is the number of tuples. Then, considering a missing value x_{ij} (the value of attribute A_j in tuple $\boldsymbol{x}_i$ is missing), we utilize the attention mechanisms to implement the action of selectively concentrating on fewer tuples for the value on attribute A_j while ignoring the others when imputing x_{ij}. Intuitively, tuples that are more similar to $\boldsymbol{x}_i$ on attribute A_j deserve more attention, thus we have

$$x'_{ij} = \sum_h s(\boldsymbol{x}_i, \boldsymbol{x}_h)_j \times x_{hj}. \quad (1)$$

Here, $s(\boldsymbol{x}_i, \boldsymbol{x}_h)_j$ is the value in row i and column h of $\mathcal{A}^j$, representing the similarity between x_{ij} and x_{hj}. Notice that x_{ij} is missing, so the question here is how to calculate the similarity to a missing value. Actually, we can estimate the similarity of two tuples on attribute A_j by comparing their similarity on other attributes, and the more correlated an attribute is to A_j, the more effective it is in estimating the similarity on attribute A_j.

The next question is how to calculate the correlation between two attributes. We build a symmetric correlation matrix $\boldsymbol{C}$ while each element c_{ij} represents the correlation between attribute A_i and A_j. For two attributes A_i and A_j, we set $c_{ij} = 0$ if $i = j$. Otherwise, we pick all the tuples with no missing values on both A_i and A_j and quantify the strength of correlation between them (denoted by c_{ij} and also c_{ji}) by the absolute value of *Cramér's V coefficient* [15].

Then, given two tuples $\boldsymbol{x}_i$ and $\boldsymbol{x}_h$, the similarity between x_{ij} and x_{hj} is measured with the weighted sum of the similarity of these two tuples on the other attributes and the weights are determined by the strength of correlation with A_j. That is,

$$s(\boldsymbol{x}_i, \boldsymbol{x}_h)_j = \sum_z c_{zj} \times sim(x_{iz}, x_{hz}). \tag{2}$$

The similarity $sim(x_{iz}, x_{hz})$ is defined as follows. If A_z is a categorical attribute, $sim(x_{iz}, x_{hz}) = 1$ when x_{iz} and x_{hz} have the same value, otherwise $sim(x_{iz}, x_{hz}) = 0$. If A_z is a numerical attribute, $sim(x_{iz}, x_{hz}) = 1 - |\frac{x_{iz} - x_{hz}}{\max_z - \min_z}|$ where $\max_z$ and $\min_z$ are the maximum and minimum values of attribute A_z respectively. Then, each row $\mathcal{A}^j_i$ of the attention map $\mathcal{A}^j$ will be

$$\mathcal{A}^j_i = Softmax(s(\boldsymbol{x}_i, \boldsymbol{x}_1)_j, \ldots, s(\boldsymbol{x}_i, \boldsymbol{x}_{n-1})_j, s(\boldsymbol{x}_i, \boldsymbol{x}_n)_j). \tag{3}$$

Example 4.1. Suppose we attempt to estimate the missing value t_2[Salary] in Table 1. We first compute the correlations between Salary (*i.e.,* A_7) and other attributes as $\mathbf{c}_7 = (0, 0.03, 0, 0.05, 0.03, 0.04, 0, 0.62)$. Then, the similarity between t_2 and t_3 on Salary is computed as $s(\boldsymbol{x}_2, \boldsymbol{x}_3)_7 = 0 \times 0 + 0.03 \times 1 + 0 \times 1 + 0.05 \times 1 + 0.03 \times 0 + 0.62 \times (1 - |(2.50 - 2.34)/(2.58 - 1.38)|) \approx 0.62$. Similarly, we can obtain $s(\boldsymbol{x}_2, \boldsymbol{x}_1)_7 \approx 0.21$. Thus, t_3 has a higher attention score than t_1 when imputing t_2[Salary], since it has more similar values on the attributes (*e.g.,* Rate) which are more relevant to Salary.

Remark. (1) Equation 1 only applies to the case where x_{ij} is missing but x_{hj} is not. When x_{hj} is also missing, $s(\boldsymbol{x}_i, \boldsymbol{x}_h)_j = 0$. When x_{ij} is not missing, we will utilize $\boldsymbol{M}$ to mask out x'_{ij} as Equation 4 (there is no need to utilize the attention mechanisms for an observed value). (2) When traversing the other attributes A_z of tuples $\boldsymbol{x}_i$ and $\boldsymbol{x}_h$ in Equation 2, if x_{iz} or x_{hz} is missing, we skip their similarity computation on attribute A_z as the above example shows. (3) We utilize a threshold κ to set the values less than κ in $\mathcal{A}^j_i$ to 0, which could help to prevent the decision bias caused by data imbalance.

Let $\boldsymbol{X}_j$ be the projection of $\boldsymbol{X}$ on attribute A_j, and $\boldsymbol{X}^{\mathcal{A}}$ is the concatenation of the product $\mathcal{A}^j \times \boldsymbol{X}_j$ across all attributes. We can then reconstruct $\boldsymbol{X}$ by

$$\boldsymbol{X}' = (\mathbf{1} - \boldsymbol{M}) \odot \boldsymbol{X}^{\mathcal{A}} + \boldsymbol{M} \odot \boldsymbol{X} \tag{4}$$

where $\odot$ denotes element-wise multiplication and $\mathbf{1}$ is all-ones matrix. Note that we only use attention mechanisms during the data pre-processing phase and not after each iteration. This is because

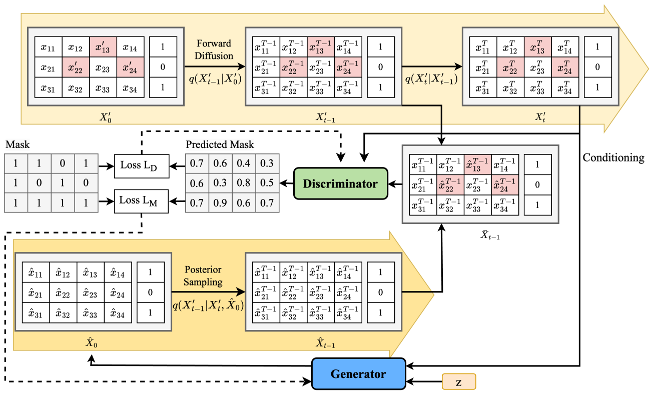


Figure 2: The Training Process of Data Imputation

the cost of calculating similarities per iteration increases significantly with large datasets but does not significantly enhance the final results.

Generator. After obtaining $\boldsymbol{X}'$, we further impute the missing values with the use of the denoising diffusion GAN, which consists of a forward diffusion process and a posterior sampling process, as presented in Figure 2. To ease the following description, we denote $\boldsymbol{X}'$ by $\boldsymbol{X}'_0$, which is the output of the attention mechanism and the input of the following diffusion model.

In the forward diffusion process, we iteratively add Gaussian noise to $\boldsymbol{X}'_0$ over T steps. In each step, we have:

$$q(\boldsymbol{X}'_t|\boldsymbol{X}'_{t-1}) = \mathcal{N}(\boldsymbol{X}'_t; \sqrt{1-\beta_t}\boldsymbol{X}'_{t-1}, \beta_t \mathbf{I}). \tag{5}$$

Here, $\mathcal{N}$ represents the Gaussian distribution; t denotes any step between 1 and T; β_t is the diffusion rate, representing the degree of noise added at step t; $\boldsymbol{X}'_t$ denotes the data achieved by introducing noise into $\boldsymbol{X}'_0$ over t steps; $\mathbf{I}$ denotes the identity matrix of the same dimension as $\boldsymbol{X}'_t$.

In the posterior sampling process, we employ a generator G to perform denoising, *i.e.,* approximating $q(\boldsymbol{X}'_{t-1}|\boldsymbol{X}'_t)$. However, instead of taking $\boldsymbol{X}'_t$ as input and outputting $\boldsymbol{X}'_{t-1}$, the generator will try to estimate $\boldsymbol{X}'_0$ and then adds noise to obtain $\boldsymbol{X}'_{t-1}$. In this way, G has the singular goal of generating $\boldsymbol{X}'_0$, rather than a series of $\boldsymbol{X}'_{t-1}$ for different step t, which is beneficial to the effective training of G.

Concretely, let $\hat{\boldsymbol{X}}_0$ denote the actual output of G to be as similar as $\boldsymbol{X}'_0$ in the observed data. We inject the noise into $\hat{\boldsymbol{X}}_0$ again to get $\hat{\boldsymbol{X}}_{t-1}$:

$$q(\hat{\boldsymbol{X}}_{t-1}|\boldsymbol{X}'_t, \hat{\boldsymbol{X}}_0) = \mathcal{N}(\hat{\boldsymbol{X}}_{t-1}; \tilde{\mu}_t(\boldsymbol{X}'_t, \hat{\boldsymbol{X}}_0), \tilde{\beta}_t \mathbf{I}) \tag{6}$$

where $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$, $\tilde{\mu}_t(\boldsymbol{X}'_t, \hat{\boldsymbol{X}}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\hat{\boldsymbol{X}}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{X}'_t$ and $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$. And we define the imputed table $\bar{\boldsymbol{X}}_{t-1}$ in step $t-1$ to be the combination of the partial observations from $\boldsymbol{X}'_{t-1}$ and imputed data from $\hat{\boldsymbol{X}}_{t-1}$:

$$\bar{\boldsymbol{X}}_{t-1} = (\mathbf{1} - \boldsymbol{M}) \odot \hat{\boldsymbol{X}}_{t-1} + \boldsymbol{M} \odot \boldsymbol{X}'_{t-1} \tag{7}$$

Subsequently, we use $\bar{\boldsymbol{X}}_{t-1}$ as the input to G and obtain a new $\hat{\boldsymbol{X}}_0$. Then we add noise to $\hat{\boldsymbol{X}}_0$ to generate $\hat{\boldsymbol{X}}_{t-2}$ by Equation 6, and derive $\bar{\boldsymbol{X}}_{t-2}$ using Equation 7. This process continues until we generate the final imputed table $\bar{\boldsymbol{X}}_0$ (*i.e.,* $\bar{\boldsymbol{X}}$).

Discriminator. To make the fake (imputed) values in $\bar{\boldsymbol{X}}$ to be as compatible as possible with the real (observed) values, some methods [2] [51] introduce a discriminator D to distinguish which components of $\bar{\boldsymbol{X}}$ are real or fake. D serves as an adversary to the generator G, ensuring that the generated data closely resembles the real data. While these approaches can effectively enhance the performance of G, the potential pitfall is that D will most likely collapse to a few states since the real value it sees is fixed each time.

In order to make G better approximate $q(\boldsymbol{X}'_{t-1}|\boldsymbol{X}'_t)$ while increasing the sample diversity, in our framework, D takes $\boldsymbol{X}'_t$, $\bar{\boldsymbol{X}}_{t-1}$ and t as inputs, identify whether each attribute value $\bar{x}_{ij}$ come from $\boldsymbol{X}'_{t-1}$ or $\hat{\boldsymbol{X}}_{t-1}$, equivalently predicting $\boldsymbol{M}$. In this way, D does not merely consider individual positive samples but their noisy versions, making it easier to grasp the data distribution and counteract model overfitting.

Objective. Let's summarize the objective functions of the generator G and discriminator D. The output of G at each step is $\hat{\boldsymbol{X}}_0$, to be as similar as $\boldsymbol{X}'_0$ in terms of the observed data. Therefore, G needs to minimize the reconstruction loss $\mathcal{L}_R$:

$$\mathcal{L}_R = \mathbf{M} \odot R(\hat{\boldsymbol{X}}_0, \boldsymbol{X}'_0) \tag{8}$$

where $R(\cdot)$ represents the calculation of mean squared error (MSE) loss for numerical data and cross-entropy loss for categorical data.

As adversarial training, G is to make D unable to distinguish which components of $\bar{\boldsymbol{X}}_{t-1}$ are real in each step, which ensures the consistency of the distributions of $p_\theta(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t)$ and $q(\boldsymbol{X}_{t-1}|\boldsymbol{X}_t)$. Therefore, G should minimize the loss $\mathcal{L}_M$:

$$\mathcal{L}_M = -\mathbb{E}_{\bar{\boldsymbol{X}}_t, t, \boldsymbol{M}}\left[(\mathbf{1} - \boldsymbol{M})\log(D(\bar{\boldsymbol{X}}_t, t))\right]. \tag{9}$$

For $m_{ij} = 1$, the corresponding loss is 0. And for $m_{ij} = 0$, minimizing $\mathcal{L}_M$ is to incentivize the discriminator to predict it as 1, *i.e.,* the quality of the imputed data is sufficient to pass off as real.

In each step t, we train D to maximize the probability of correctly predicting $\boldsymbol{M}$. Consequently, we define the loss $\mathcal{L}_D$ for the discriminator as:

$$\mathcal{L}_D = -\mathbb{E}_{\bar{\boldsymbol{X}}_t, t, \boldsymbol{M}}\left[\boldsymbol{M}\log \mathrm{D}(\bar{\boldsymbol{X}}_t, t) + (\mathbf{1} - \boldsymbol{M})\log(\mathbf{1} - \mathrm{D}(\bar{\boldsymbol{X}}_t, t))\right]. \tag{10}$$

Minimizing $\mathcal{L}_D$ means minimizing the difference between the predicted mask $\mathrm{D}(\bar{\boldsymbol{X}}_t, t)$ and the ground truth $\boldsymbol{M}$, which is predetermined by $\boldsymbol{X}$.

5 DATA UTILITY PLUG-IN

If the imputed relational table $\bar{\boldsymbol{X}}$ is taken as training data, the data utility plug-in can be utilized to ensure the applicability of the imputed data for subsequent classification tasks. We introduce a *Learner* L, which acts as the classifier used in the subsequent tasks. Thus, we expect the imputed data $\bar{\boldsymbol{X}}$ to minimize the learner's classification loss on the given validation set. This validation loss is denoted by $\mathcal{L}_L$ in Figure 1, which affects the update of the generator's parameters via backward propagation to generate the expected imputed data.

Suppose the downstream classifier and the validation data are available. We can directly employ them in this plug-in: training the downstream classifier while imputing the missing values in the training data. Otherwise, we construct the learner L using a two-layer fully connected neural network, and the validation data will come from $\bar{\boldsymbol{X}}$. Specifically, we randomly take a portion of the tuples from which all of its attribute values are observed in $\boldsymbol{X}$ as the validation set $\boldsymbol{X}_v$, and the rest of the tuples in $\bar{\boldsymbol{X}}$ as the training set $\boldsymbol{X}_\eta$ to train L. The validation loss is determined by the difference between the labels obtained by applying L to the validation set and the true labels:

$$\mathcal{L}_L = \frac{1}{|\boldsymbol{X}_v|}\sum_{i=1}^{|\boldsymbol{X}_v|} H(y_i, p_i) \tag{11}$$

where $H(\cdot)$ denotes the cross-entropy loss, and y_i, p_i is the ground truth and predicted label of $\bar{x}_i$ respectively.

$\mathcal{L}_L$ is derivable with respect to the parameters θ_G of the generator G. According to the chain rule, we have

$$\frac{\partial \mathcal{L}_L}{\partial \theta_G} = \frac{\partial \mathcal{L}_L}{\partial \theta_L} \cdot \frac{\partial \theta_L}{\partial \boldsymbol{X}_\eta} \cdot \frac{\partial \boldsymbol{X}_\eta}{\partial \theta_G}. \tag{12}$$

Here θ_L denote the parameter of L. That is, we can teach G to generate the training data (*i.e.,* impute the missing values in the training data) that enables the learner to perform well on the validation set (the observed data). If the distribution of $\boldsymbol{X}$ and test data in the target task is similar (an assumption often made when applying machine learning models), we can guarantee the utility of our imputed data for training the downstream model. Meanwhile, the learner can contribute to keeping the same data distribution between the imputed and observed data.

Several considerations arise in the design of this plug-in. First, it will be costly to continuously train a new classifier for each generated data instance. As a result, we incorporate $\mathcal{L}_L$ only after the diffusion process is completely finished. Second, in the case of a high rate of missing data, it becomes difficult to find tuples that do not contain missing values for the validation set. In this situation, we utilize the discriminator D to construct the most appropriate validation set. Concretely, we assign a confidence score to each tuple in $\bar{\boldsymbol{X}}$: for each observed attribute value, it has a confidence score of 1, whereas each missing attribute value receives a score derived from D, *i.e.,* the probability that this value is enough to pass off as real; the confidence score for a tuple is the sum of scores across all its attributes. We then select some top of the tuples after sorting in descending order of their confidence score to compose the validation set.

6 DATA DEPENDENCY PLUG-IN

In this section, we introduce our data dependency plug-in, which extracts potential FDs from $\boldsymbol{X}$ with missing values and encodes these constraints into the training process of the data imputation module through one extra loss $\mathcal{L}_F$. Moreover, we will simultaneously impute missing values and repair inaccurate constraints based on the relative confidence between data and FDs.

6.1 Discovery of Functional Dependencies

Berti-Équille et al. [4] have summarized previous work for discovering FDs from data with missing values and mentioned their respective drawbacks. For example, using NULL semantics or directly skipping all tuples with missing values will make many correct FDs undiscoverable. To avoid this, given an FD $\varphi : Z \rightarrow \{A\}$, we expect to verify the soundness of φ using all tuples that do not contain missing values on $Z \cup A$, rather than skipping all tuples with missing values on any of the attributes U.

We adapt TANE [20] to achieve this goal. TANE introduces the notation of a *partition*, denoted by π. For a set of attributes Z, π_Z is a collection of *equivalence classes* of tuples, such that each class has a unique value for the attribute set Z. We redefine the partition. Specifically, given a set of attributes Z, we omit the missing data to calculate π_Z. Thus, the union of these equivalence classes in π_Z equals a set of tuples with no missing value in Z. For example, $\pi_{\text{AreaCode}} = \{\{t_1, t_2, t_3, t_4\}, \{t_6, t_7\}, \{t_8\}\}$. Then, we redefine the formulation for deriving $\pi_{Z\cup\{A\}}$ from π_Z and $\pi_{\{A\}}$ under the new definition of partition:

$$\pi_{Z\cup\{A\}} = (\cup_{z_i \in \pi_Z}(z_i \cap (\cup_{a_j \in \pi_{\{A\}}} a_j))) \setminus \{\emptyset\} \tag{13}$$

This remove tuples from $\pi_{Z\cup\{A\}}$ that contain missing values on attribute Z or A. Under the above setting, an FD $Z \longrightarrow A$ holds if and only if for each $z_i \in \pi_Z$, there exist at most one subset of z_i in $\pi_{Z\cup\{A\}}$.

We set a threshold τ to limit the maximum number of attributes probed on the left-hand side of an FD. The reason is that when considering an excessive number of attribute combinations, the FD holds mostly because there are no two tuples with the same value in attributes on the left-hand side, which does not work for reasoning about missing values. Moreover, we will prune some of the candidate FDs during the mining process. For example, if $Z \longrightarrow A$ holds, we will not consider $Z \cup Z' \longrightarrow A$ unless $Z \longrightarrow A$ is found to be invalid in the following phase of refinement. Thus, the number of potential FDs mined from the incomplete tabular data is not particularly large. This approach might inadvertently lead to the discovery of some erroneous FDs. The set F of potential FDs mined from $\boldsymbol{X}$ may differ from those $F^{\checkmark}$ held in $\boldsymbol{X}^{\checkmark}$. Thus, they will be refined based on the imputed data in the subsequent process, illustrated in Section 6.3, where most of the erroneous FDs can be rectified in a few iterations.

6.2 Modeling of Functional Dependencies

Given a set F of potential FDs mined from $\boldsymbol{X}$, we build a dependency model Φ_φ for each FD $\varphi \in F : Z \longrightarrow A$, which takes the attribute values of Z as input and predicts the attribute value of A. Thus, we utilize all tuples with no missing value in $Z \cup \{A\}$ as training data for Φ_φ, and for each tuple, their attribute values of Z and A are taken as features and the label, respectively. Actually, these dependency models could also be pre-trained using the master data or knowledge bases after alignment. This allows more external information to guide the imputation of missing values.

It is worth noting that the dependency model Φ_φ actually accepts values in all attributes $U \setminus \{A\}$ as input, not just in Z. But all attribute values except Z are set to zero for input to the model. That means we manually perform a "dropout" operation that ignores the inputs in the other attributes $U \setminus (Z \cup \{A\})$, but still retains the nodes on the input layer of Φ_φ. The reason behind this is that as the imputation becomes more accurate, φ might be proven to be incorrect. Thus we need to add more attributes to the left-hand side of φ to make it valid (more details can be found in Section 6.3). In such a case, we need to "activate" the corresponding input nodes and fine-tune the model instead of training a new dependency model, which will save a lot of training costs without sacrificing the model's performance.

Then, given the imputed table $\bar{\boldsymbol{X}}$, we expect it to satisfy all the FDs in F. That is, the values imputed by the generator G are desired to be consistent with the predictions of the dependency models. Let $\mathbf{R}_F = \{A_{\varphi_1}, \cdots, A_{\varphi_{|F|}}\}$ where A_{φ_i} be the attribute in the right-hand side of the ith FD in F. Then, we define the following loss for the generator:

$$\mathcal{L}_F = \sum_{i=1}^{n} \sum_{j=1}^{|F|} \left\| \bar{\mathbf{v}}_{ij} - \mathbf{v}_{ij} \right\|_1 \tag{14}$$

where $||\cdot||_1$ denotes the L1-norm as the distance measure between two vectors; $\bar{\mathbf{v}}_{ij}$ and $\mathbf{v}_{ij}$ are representations of the attribute value of the ith tuple in attribute A_{φ_j} given by the generator G and corresponding dependency model Φ_{φ_j}. In fact, since we take one-hot encoding for the categorical attributes in the data imputation module, $\bar{\mathbf{v}}_{ij}$ and $\mathbf{v}_{ij}$ are predicted probability vectors given by the model G and Φ_{φ_j} respectively.

6.3 Inaccurate FDs Refinement

When mining the potential functional dependencies, we ignore the missing data, assuming that tuples containing missing values always satisfy the mined FDs, which is clearly flawed. As the imputation becomes more accurate, we must go back to determine whether these FDs are valid. In this section, we introduce a framework to refine inaccurate FDs based on the confidence of data and constraints.

Confidence of Tuple. The confidence of a tuple is determined by the confidence of its attribute values. Given an FD $\varphi : Z \longrightarrow A$, the confidence of a tuple $\bar{\boldsymbol{x}}_i$ under φ is the minimum value of the confidence of attribute values in $Z \cup \{A\}$:

$$c_\varphi(\bar{\boldsymbol{x}}_i) = min(c(\bar{x}_{ij})), A_j \in Z \cup \{A\} \tag{15}$$

Confidence of attribute value. It goes without saying that the confidence of observed data should be 1. As for the missing data, we derive the confidence of imputed value $\bar{x}_{ij}$ based on the predictions of generator G and discriminator D. As stated earlier, we adopt one-hot encoding for the categorical attribute. Therefore, for the jth attribute of the ith tuple, the output of G is a predicted probability vector, and $\bar{x}_{ij}$ is the value with the highest probability, which is denoted by $p_G(x_{ij})$ and can also be taken as the confidence of imputed value $\bar{x}_{ij}$ given by G. D gives the probability $p_D(x_{ij})$ that $\bar{x}_{ij}$ turns out to be observed. The larger $p_D(x_{ij})$, the more $\bar{x}_{ij}$ looks like being real. We have also found from the experiment that $p_D(x_{ij})$ being larger means x_{ij} is imputed correctly. Thus, we take it as the confidence of imputed value $\bar{x}_{ij}$ provided by D. In brief, the confidence of $\bar{x}_{ij}$ is the weighted sum of $p_G(x_{ij})$ and $p_D(x_{ij})$:

$$c(\bar{x}_{ij}) = \begin{cases} 1, & m_{ij} = 1 \\ \frac{w_G \times p_G(x_{ij}) + w_D \times p_D(x_{ij})}{w_G + w_D}, & m_{ij} = 0 \end{cases} \tag{16}$$

Here w_G and w_D are the accuracy of G and D measured from the observed data. Concretely, w_G is the proportion of observed values G correctly imputed among all observed values. The more values G can correctly impute, the more reliable the probability $p_G(x_{ij})$ it gives. w_D is the sum of the predicted probabilities over the observed values divided by the total number of the observed values. The larger w_D is, the better D is at recognizing observed values, and the more reliable the probability $p_D(x_{ij})$ it gives.

Example 6.1. Given the FD φ_3 : AreaCode→SingleExemp, let us compute the confidence of t_5 under φ_3. Suppose the predicted probability vector given by G for t_5[AreaCode] is (0.05, 0.8, 0.15), and the output from D is 0.9. There are totally 42 observed data in all categorical attributes. Let us say G correctly predicts 32 of them, and the sum of predicted probabilities by D is 38.64. We have $w_G = 32/42 \approx 0.76$ and $w_D = 38.64/42 \approx 0.92$. The confidence of $t_5[Areacode]$ is $(0.76 \times 0.8 + 0.92 \times 0.9)/(0.76 + 0.92) \approx 0.85$. Subsequently, we can determine the confidence of t_5 under φ_3 as $\min(0.85, 1) = 0.85$.

Confidence of FD. Following the previous work [4], we construct a subset of tuples $\bar{\boldsymbol{X}}_\varphi$ from $\bar{\boldsymbol{X}}$ that satisfy φ to measure the confidence $c(\varphi)$ of an FD $\varphi : Z \longrightarrow A$. Berti-Équille et al. [4] continue the frequency-based strategy (the more frequently a value occurs, the more likely it is to be correct) while ignoring each tuple's confidence. To avoid using erroneous tuples to measure the confidence of an FD, we choose tuples with higher confidence, which is consistent with the intuition that the higher the confidence of tuples that satisfy a constraint, the more likely the constraint is correct.

$\bar{\boldsymbol{X}}_\varphi$ should initially contain all tuples with no missing value in attributes $Z \cup \{A\}$ (The FD φ is extracted from these tuples, so they are naturally supposed to serve as evidence that φ is valid). For the other tuples, we sort them in descending order of their confidence and add them in turn to $\bar{\boldsymbol{X}}_\varphi$. Note that if a tuple has already been in $\bar{\boldsymbol{X}}_\varphi$, tuples that have violation with it *w.r.t.* φ should be excluded. Finally, we have

$$c(\varphi) = \frac{\sum_{\bar{\boldsymbol{x}}_i \in \bar{\boldsymbol{X}}_\varphi} c_\varphi(\bar{\boldsymbol{x}}_i)}{\sum_{\bar{\boldsymbol{x}}_j \in \bar{\boldsymbol{X}}} c_\varphi(\bar{\boldsymbol{x}}_j)}. \tag{17}$$

It is worth noting that tuples with unique values in attributes Z are not used to derive the confidence of φ since they do not play any role in determining whether the FD holds. Thus, we remove them from $\bar{\boldsymbol{X}}_\varphi$ and $\bar{\boldsymbol{X}}$ before calculating the confidence of rule φ.

Example 6.2. Continue to consider φ_3 in Example 6.1. Suppose t_5[Areacode] and t_3[SingleExemp] are correctly imputed with "501" and "1000" respectively, and the confidence of t_3 is 0.67. We can find from the imputed table that t_3 have the same value with $\{t_1, t_2, t_4\}$ in AreaCode but different in SingleExemp. Given that $\{t_1, t_2, t_4\}$ under φ_3 are all observed, we put $\{t_1, t_2, t_4\}$ in $\bar{\boldsymbol{X}}_{\varphi_3}$. Similarly, $\{t_6, t_7\}$ are also added to $\bar{\boldsymbol{X}}_{\varphi_3}$. We exclude t_8 since it has a distinct value of AreaCode. Then, the confidence of φ_3 is calculated as $c(\varphi_3) = (1+1+1+1+1)/(1+1+0.67+1+0.85+1+1) \approx 0.77$.

FD Refinement. We compare the confidence of a constraint and tuples violating it to determine whether an FD φ is valid, *i.e.,* $\varphi \in F^{\checkmark}$. Concretely, the confidence of φ can be calculated by Equation 17. In the meantime, we choose the tuple $\bar{\boldsymbol{x}}^*_\varphi$ from $\bar{\boldsymbol{X}} \setminus \bar{\boldsymbol{X}}_\varphi$ with the highest confidence as an indicator that φ may be invalid. It stems from the notion that higher confidence indicates the tuple $\bar{\boldsymbol{x}}^*_\varphi$ is closer to its true value, and φ obviously cannot hold on $\bar{\boldsymbol{X}}_\varphi \cup \{\bar{\boldsymbol{x}}^*_\varphi\}$. When the FD φ has higher confidence than $\bar{\boldsymbol{x}}^*_\varphi$, we continue to employ it to guide the imputation of missing values. Otherwise, φ is considered inaccurate and necessitates a refinement.

<u>*Remark.*</u> Here, we can find the calculation of FD confidence in Equation 17 indirectly improves the confidence of the violated tuples required for the refutation of FD, thus ensuring that only a correct pair of tuples conflicting on the FD can give the conclusion that it is inaccurate.

Algorithm 1: Training Process of DDGANI

```
Input: relational table $\boldsymbol{X}$ with missing values, mask $\boldsymbol{M}$, the
       number of diffusion steps $T$, input noise $z$,
       hyper-parameters such as loss weights $\alpha, \beta$
Output: imputed relational table $\bar{\boldsymbol{X}}$
1  $\boldsymbol{X}' \longleftarrow$ pre-processing $\boldsymbol{X}$ with the self-attention mechanism;
2  initialize the parameters of generator G and discriminator D;
3  extract potential FDs $F$ from $\boldsymbol{X}$ and build their
     corresponding dependency models;
4  repeat
5      $\{\boldsymbol{X}'_1, \cdots, \boldsymbol{X}'_T\} \longleftarrow$ add noise to $\boldsymbol{X}'$ over $T$ steps;
6      for step $t \in [T, \cdots, 1]$ do
7          $\hat{\boldsymbol{X}}_0 \longleftarrow G(\boldsymbol{X}'_t, z, t)$;
8          compute the reconstruction loss $\mathcal{L}_R$ with $\hat{\boldsymbol{X}}_0$ and $\boldsymbol{X}'$;
9          $\bar{\boldsymbol{X}}_0 \longleftarrow (\boldsymbol{1} - \boldsymbol{M}) \odot \hat{\boldsymbol{X}}_0 + \boldsymbol{M} \odot \boldsymbol{X}'$;
10         compute the data dependency loss $\mathcal{L}_F$ with $\bar{\boldsymbol{X}}_0$;
11         $\hat{\boldsymbol{X}}_{t-1} \longleftarrow$ posterior sampling with $\hat{\boldsymbol{X}}_0$ and $\boldsymbol{X}'_t$;
12         $\bar{\boldsymbol{X}}_{t-1} \longleftarrow (\boldsymbol{1} - \boldsymbol{M}) \odot \hat{\boldsymbol{X}}_{t-1} + \boldsymbol{M} \odot \boldsymbol{X}'_{t-1}$;
13         $\bar{\boldsymbol{M}} \longleftarrow D(\boldsymbol{X}'_t, \bar{\boldsymbol{X}}_{t-1}, t)$;
14         compute the discrimination loss $\mathcal{L}_D$ and $\mathcal{L}_M$;
15         $\mathcal{L}_G \longleftarrow \mathcal{L}_R + \alpha\mathcal{L}_M + \beta\mathcal{L}_F$;
16         update the parameters of G, D by $\mathcal{L}_G$, $\mathcal{L}_D$
             respectively;
17     refine $F$ based on $\bar{\boldsymbol{X}}_0$ and retrain the dependency models;
18     $\bar{\boldsymbol{X}}_v \longleftarrow$ randomly pick some tuples from $\bar{\boldsymbol{X}}_0$ whose
         attribute values are all observable;
19     train the learner L with the rest tuples $\bar{\boldsymbol{X}}_0 \setminus \bar{\boldsymbol{X}}_v$;
20     evaluate L on $\bar{\boldsymbol{X}}_v$ and compute $\mathcal{L}_L$;
21     update the parameters of G by $\mathcal{L}_L$;
22 until convergence of G and D;
23 return $\bar{\boldsymbol{X}}_0$
```

Before discussing specific strategies of FD refinement, we first analyze what changes F has to make to become $F^{\checkmark}$. There are four optional steps: (1) add a new FD φ' that $\varphi' \in F^{\checkmark}$; (2) remove an FD φ from F that $\varphi \notin F^{\checkmark}$; (3) add some new attributes to the left-hand side of $\varphi \in F$; (4) remove some attributes from the left-hand side of $\varphi \in F$. Choice 1 is only required if the missing rate is very high (experimentally found to be >85%). This is uncommon and what we can do is re-mining the FDs from $\bar{\boldsymbol{X}}$. Thus, we will not discuss this scenario in detail. Choice 4 will never be adopted, and that is because if φ does not hold, then it cannot be valid after removing an attribute from the left-hand side of φ. Hence, we only consider choice 2 and 3, *i.e.,* when the confidence of an FD is less than the violated tuple, we add some attributes to its left-hand side. If the number of attributes in the left-hand side after appending exceeds the threshold τ, we discard this FD when it still does not hold.

How to repair an FD by adding more attributes to the left-hand side is not a new problem and has been studied in previous work such as [12]. The difference is that we will try to find all possible solutions for appending attributes. For example, upon discovering that Areacode, MaritalStatus→SingleExemp is one way to fix the erroneous FD Areacode→SingleExemp, we will not stop there and continue to find another way to refine it such as Areacode, MarriedExemp→SingleExemp. This allows us to dig into more dependencies between attributes, which can help with the missing value imputation.

In fact, we could adopt some simple sorting strategies here. When appending attributes to $\varphi : Z \rightarrow A$, we need the power set of $U \setminus (Z \cup \{A\})$ (excluding $\emptyset$ and sets larger than $\tau - |Z|$). Each item in the power set is taken in ascending order of the size into consideration whether it can be appended to the left-hand side of φ. Here, if appending an attribute set Z' yields an FD $\varphi' : Z \cup Z' \rightarrow A$, and $c(\varphi') \geq c(\bar{\mathbf{x}}^*_{\varphi'})$, then no more super sets of Z' are considered. We can also choose to combine the correlation of attributes but without associated theoretical guarantees: if appending Z' does not result in a valid FD $\varphi' : Z \cup Z' \rightarrow A$, we will not consider other candidates that have a weaker correlation with A than Z' referring to our correlation matrix $\boldsymbol{C}$. We discard φ if it still cannot hold after trying all the possible appends.

Retraining of Dependency Model. If more attributes are added to the left-hand side of FD φ (φ is refined to $\varphi' : Z \cup Z' \rightarrow A$), we need to retrain the dependency model Φ_φ after enabling the input nodes of attributes Z' to obtain a new model $\Phi_{\varphi'}$. That is, we utilize all tuples in $\bar{\boldsymbol{X}}_{\varphi'}$ to retrain Φ_φ: for each tuple, their attribute values of $Z \cup Z'$ and A are taken as features and the label, respectively, and all attribute values except $Z \cup Z'$ still need to be set to zero for input to the model. Since Φ_φ has already captured some inner connection between Z and A, the retraining mainly goes to determine the effect of the newly added attributes on A, which will not take long. If the FD φ is entirely discarded, the corresponding dependency model Φ_φ is also eliminated with it.

7 ENTIRE TRAINING PROCESS OF DDGANI

The overall workflow of DDGANI is presented in Algorithm 1. Let us first focus on the data imputation module, which is in black. The missing data is initially pre-processed with the attention mechanism to obtain $\boldsymbol{X}'$ (line 1). We then initialize the parameters of generator G and discriminator D (line 2) and train the models (lines 4-22). A forward diffusion process is applied on $\boldsymbol{X}'$ in the beginning of each epoch to produce $\{\boldsymbol{X}'_1, \cdots, \boldsymbol{X}'_T\}$ (line 5). Then in each step t of denoising, G takes $\boldsymbol{X}'_t$, t and a random Gaussian noise z as input and outputs $\hat{\boldsymbol{X}}_0$ (line 7). The loss $\mathcal{L}_R$ is computed based on the differences between $\hat{\boldsymbol{X}}_0$ and $\boldsymbol{X}'$ (line 8). Then, a posterior sampling process derives $\hat{\boldsymbol{X}}_{t-1}$ from $\hat{\boldsymbol{X}}_0$ and $\boldsymbol{X}'_t$ (line 11). We integrate $\hat{\boldsymbol{X}}_{t-1}$ and $\boldsymbol{X}'_{t-1}$ to obtain the final imputed table $\bar{\boldsymbol{X}}_{t-1}$ at step $t-1$, which is fed into D with $\boldsymbol{X}'_t$ and $t-1$ to predict the mask $\boldsymbol{M}$ (lines 12-13). The differences between the predicted result and $\boldsymbol{M}$ will be used to compute the loss $\mathcal{L}_D$ and $\mathcal{L}_M$, which are used to update the parameters of D and G along with $\mathcal{L}_R$, respectively (lines 14-16).

Then, let us look at the usage of the data utility plug-in, which is in gray. After the data imputation module has finished its work, we randomly draw some tuples without missing values from $\bar{\boldsymbol{X}}_0$ as the validation set $\bar{\boldsymbol{X}}_v$, and a learner L is trained using the remaining data (lines 18-19). $\mathcal{L}_L$ is the validation loss of L, which will be used to update the parameters of G via meta-learning paradigm (lines 20-21).

The data dependency plugin is required to be integrated for use in the data imputation module, which is in brown. We will first mine potential FDs at the beginning and build their corresponding dependency models (line 3). After obtaining $\hat{\boldsymbol{X}}_0$ at each step t, we combine $\hat{\boldsymbol{X}}_0$ and $\boldsymbol{X}'$, yielding the imputed data $\bar{\boldsymbol{X}}_0$ and calculating the loss $\mathcal{L}_F$ (lines 9-10). This loss will be added into $\mathcal{L}_G$ to update the parameters of G (line 15). These FDs may be refined according to the confidence of tuples and constraints after the missing value imputation (line 17).

Table 2: Dataset Description

Dataset	#Instance	#Numerical	#Categorical	#FDs
Wine	178	13	0	0
Wireless	2000	7	0	0
Spam	4601	57	0	0
Tic-Tac-Toe	958	0	9	0
Flare	1066	0	12	0
Balance	625	0	4	0
Adult	30148	5	9	2
Hospital	1000	2	7	8
Tax	100000	2	7	9
Thoracic	470	3	13	0
Contraceptive	1473	2	7	0
Australian	690	6	8	0
Beers	2410	3	3	0

8 EXPERIMENTS

8.1 Experimental Settings

Dataset. We evaluate our approach on thirteen publicly available datasets, including three datasets with all numerical attributes *Wine*, *Wireless*, *Spam*, three datasets with all categorical attributes *Tic-Tac-Toe*, *Flare*, *Balance*, and seven datasets with both numerical and categorical attributes *Adult*, *Hospital* [37], *Tax* [6], *Thoracic*, *Contraceptive*, *Australian*, *Beers*[1]. Datasets not specifically attributed are from the UCI Machine Learning Repository (https://archive.ics.uci.edu/). Table 2 presents an overview of these datasets, including the number of tuples, numerical attributes, categorical attributes, and functional dependencies. Here, the FDs are mined by TANE [20] from the complete dataset and treated as the true FDs in Section 8.6.

Each dataset is randomly divided into three parts: 70% of the training set with missing values injected to evaluate the accuracy of imputation methods, 20% of the test set which remains complete and is used to report the accuracy of the downstream classifier, and 10% of the validation set for the hyper-parameter searching.

Missing Value Injection. We remove some attribute values from the datasets completely at random with probability p (*Missing Completely at Random, MCAR*). The missing rate p is set to 20% by default, and it is changed from 10% to 50% in Section 8.3 to show the effectiveness of our proposed method under different missing rates. Section 8.4 shows the evaluation of our method under *MAR (Missing at Random)*, *MNAR (Missing Not at Random)* and Regional Missing. More details can be found in Section 8.4. Note that Beers

[1]https://www.kaggle.com/datasets/nickhould/craft-cans

Table 3: Overall Evaluation of the Model (Average ± Std)(20% Missing Rate, MCAR)

Dataset	Metrics	Mean	MissFI	GAIN	DataWig	Baran	AimNet	NOT-MIWAE	Graf	VAIM	EGG-GAE	DDGANI
Wine	ARMSE	0.296±0.000	0.151±0.003	0.171±0.007	0.163±0.005	0.225±0.013	0.241±0.014	0.169±0.004	0.297±0.014	0.182±0.013	0.150±0.004	**0.145±0.001**
	AMAE	0.250±0.000	0.113±0.002	0.135±0.007	0.129±0.004	0.153±0.010	0.185±0.010	0.128±0.004	0.251±0.012	0.143±0.010	0.120±0.003	**0.106±0.001**
Wireless	ARMSE	0.352±0.000	0.120±0.000	0.140±0.007	0.119±0.002	0.231±0.018	0.224±0.012	0.121±0.001	0.355±0.001	0.148±0.006	0.164±0.008	**0.118±0.001**
	AMAE	0.318±0.000	0.097±0.000	0.109±0.006	0.094±0.001	0.187±0.017	0.165±0.007	0.095±0.001	0.322±0.001	0.117±0.005	0.134±0.007	**0.093±0.001**
Spam	ARMSE	0.053±0.000	0.048±0.000	0.051±0.001	0.052±0.001	0.057±0.002	0.051±0.007	0.056±0.001	0.054±0.002	0.054±0.001	0.049±0.001	**0.047±0.000**
	AMAE	0.022±0.000	0.023±0.000	0.022±0.000	0.025±0.000	**0.016±0.001**	0.022±0.001	0.030±0.001	0.022±0.000	0.020±0.003	0.019±0.000	0.021±0.000
Tic-Tac-Toe	ARMSE	0.590±0.009	0.529±0.008	0.537±0.013	0.514±0.008	0.607±0.033	0.525±0.007	0.529±0.019	0.523±0.006	0.582±0.006	0.527±0.003	**0.512±0.002**
	AMAE	0.590±0.009	0.529±0.008	0.537±0.013	0.514±0.008	0.607±0.033	0.525±0.007	0.529±0.019	0.523±0.006	0.582±0.006	0.527±0.003	**0.512±0.002**
Flare	ARMSE	0.260±0.000	0.240±0.000	0.200±0.009	0.205±0.007	0.241±0.009	0.241±0.002	0.226±0.004	0.262±0.004	0.240±0.014	0.183±0.001	**0.173±0.003**
	AMAE	0.260±0.000	0.240±0.000	0.200±0.009	0.205±0.007	0.241±0.009	0.241±0.002	0.226±0.004	0.262±0.004	0.240±0.014	0.183±0.001	**0.173±0.003**
Balance	ARMSE	0.831±0.004	0.818±0.023	0.823±0.004	0.818±0.020	0.828±0.019	0.783±0.013	0.822±0.008	0.802±0.008	0.814±0.016	0.776±0.001	**0.772±0.007**
	AMAE	0.831±0.004	0.818±0.023	0.823±0.004	0.818±0.020	0.828±0.019	0.783±0.013	0.822±0.008	0.802±0.008	0.814±0.016	0.776±0.001	**0.772±0.007**
Adult	ARMSE	0.341±0.000	0.258±0.000	0.250±0.007	0.243±0.004	0.292±0.000	0.293±0.001	0.254±0.005	0.338±0.000	0.245±0.012	0.217±0.003	**0.195±0.000**
	AMAE	0.325±0.000	0.244±0.000	0.234±0.008	0.230±0.004	0.273±0.000	0.272±0.001	0.238±0.007	0.323±0.000	0.229±0.012	0.203±0.003	**0.180±0.000**
Hospital	ARMSE	0.618±0.000	0.254±0.003	0.365±0.026	0.398±0.005	0.276±0.062	0.169±0.003	0.568±0.012	0.267±0.003	0.600±0.002	0.147±0.008	**0.133±0.002**
	AMAE	0.610±0.000	0.239±0.003	0.351±0.026	0.387±0.005	0.255±0.061	0.139±0.001	0.554±0.012	0.258±0.003	0.580±0.003	0.137±0.008	**0.115±0.002**
Tax	ARMSE	0.615±0.000	0.380±0.011	0.558±0.035	0.367±0.019	0.534±0.011	0.422±0.026	0.517±0.027	0.369±0.082	0.586±0.042	0.353±0.012	**0.349±0.004**
	AMAE	0.600±0.000	0.360±0.009	0.526±0.039	0.352±0.020	0.497±0.008	0.391±0.026	0.505±0.029	0.355±0.076	0.569±0.047	0.337±0.008	**0.325±0.003**
Thoracic	ARMSE	0.170±0.005	0.161±0.010	0.175±0.004	0.154±0.001	0.142±0.014	0.198±0.001	0.154±0.006	0.166±0.008	0.153±0.005	0.151±0.001	**0.139±0.007**
	AMAE	0.158±0.005	0.151±0.009	0.163±0.003	0.146±0.000	**0.129±0.014**	0.171±0.001	0.145±0.006	0.155±0.009	0.143±0.004	0.148±0.001	0.132±0.007
Contraceptive	ARMSE	0.351±0.003	0.328±0.003	0.341±0.006	0.301±0.007	0.343±0.017	0.337±0.003	0.321±0.006	0.313±0.004	0.326±0.011	0.309±0.001	**0.290±0.001**
	AMAE	0.339±0.003	0.318±0.002	0.331±0.006	0.292±0.007	0.331±0.016	0.328±0.003	0.312±0.006	0.302±0.004	0.317±0.010	0.303±0.001	**0.281±0.001**
Australian	ARMSE	0.295±0.004	0.269±0.013	0.264±0.007	0.264±0.001	0.297±0.008	0.255±0.006	0.260±0.004	0.265±0.008	0.272±0.006	**0.235±0.000**	0.237±0.003
	AMAE	0.273±0.004	0.254±0.014	0.247±0.008	0.246±0.002	0.278±0.008	0.232±0.004	0.243±0.004	0.247±0.005	0.255±0.004	0.230±0.000	**0.226±0.002**
Beers	ARMSE	0.152±0.000	0.112±0.001	0.147±0.000	0.113±0.000	0.131±0.028	0.145±0.000	0.120±0.001	0.152±0.000	0.117±0.000	0.111±0.000	**0.104±0.001**
	AMAE	0.083±0.000	0.049±0.000	0.080±0.000	0.050±0.000	0.072±0.004	0.080±0.000	0.055±0.002	0.083±0.000	0.052±0.000	0.050±0.000	**0.047±0.001**

is a real-world dataset with 1067 missing values. Thus, we do not inject more missing values into Beers, and the ground truth is obtained through crowdsourcing techniques.

Evaluation Metrics. We employ two commonly used metrics to measure the accuracy of imputation methods: *average root mean square error* (ARMSE) [44] and *average mean absolute error* (AMAE) [35]. ARMSE calculates the average of the errors for each attribute. Specifically, it utilizes the *root mean square error* (RMSE) [22] for numerical attributes and the *accuracy error* (AR) [35] for categorical ones. The smaller ARMSE corresponds to the better imputation result. Similar to ARMSE, AMAE employs *mean absolute error* (MAE) [22] for numerical attributes and AR for categorical ones. The smaller AMAE corresponds to the better imputation result. We also measure the *accuracy* of the downstream classifier in some experiments to verify the utility of the imputed dataset.

Baseline. We compare DDGANI against the following data imputation methods. (1) *Mean* [17], which imputes the missing values by the mean of each numerical attribute and the mode of each categorical attribute; (2) *MissFI* [44], which utilizes the missing forest to impute the missing data; (3) *GAIN* [51], which employs a GAN-based model and utilizes a hinting mechanism to optimize the model for imputing missing values; (4) *DataWig* [5], which trains an imputation model for each attribute that imputes the missing values based on the observed values in all other attributes. (5) *Baran* [28], which generates potential imputations using pre-trained error corrector models and selects representative tuples for manual correction to fine-tune the corrector models. (6) *AimNet* [49], which utilizes a transformer to reconstruct missing data, also based on FDs and the attention scores between attributes. (7) *NOT-MIWAE* [21], which applies deep latent variable models (DLVMs) to the problem of missing data imputation; (8) *Graf* [36], which uses sequence GAN to learn conditional functional dependencies from the dirty data and imputes the missing values based on these learned CFDs. (9) *VAIM* [19], which utilizes a VAE-based model with attention mechanisms and learner to optimize the generative model for estimating missing values; (10) *EGG-GAE* [46], which imputes the missing values by adopting graph neural networks; We also compare with pre-trained language model based methods including *ChatGPT* and *RPT* [45], and traditional rule-based approach *Holistic* [14] for data imputation.

Hyperparameter Settings. We set the diffusion timestep T to 6, the threshold κ for the attention mechanism to 0.05, and the limit τ for the number of attributes on the left-hand side of FDs to 3. The diffusion rate β_t at step t is defined to be linearly increasing constants between $\beta_1 = 10^{-4}$ and $\beta_T = 2 \times 10^{-2}$. Other hyperparameters in DDGANI and each competing method are decided via cross-validation.

Experimental Environment. All experiments were conducted on an Intel(R) Xeon(R) Silver 4210R 2.40GHz server with NVIDIA GeForce RTX 3090 24GiB (GPU) and 94GB of RAM, running Ubuntu 20.04 system.

8.2 Overall Evaluation of the Model

We first evaluate the performance of DDGANI on thirteen datasets with 20% missing rate under the missing completely at random (MCAR) model. The results are presented in Table 3. We conduct five runs for each method and record the average and standard deviation of the experimental results. The best scores are in bold, and the second-best scores are underlined. The results of our method are on a gray background.

Table 3 shows that DDGANI outperforms other methods in most datasets. This can be attributed to the way DDGANI approaches the target distribution through a series of incremental approximations. This iterative approach focuses on refining the current distribution rather than learning a distribution from scratch, allowing the model to better capture subtle nuances and intricate dependencies within the data. Additionally, the utilization of the two plug-ins also helps the model better learn the data distribution and the data dependencies.

Furthermore, it's worth noting that GAIN, NOT-MIWAE, and VAIM all exhibit subpar performance when dealing with datasets containing a mix of numerical and categorical attributes. This is due to the following reasons: GAIN suffers from mode collapse when generating categorical data, which may result in overfitting certain observed values in the categorical attribute while ignoring the true data distribution. NOT-MIWAE and VAIM have the assumption that the latent space obeys a Gaussian distribution and generates samples from a complex distribution in one shot, which is not optimal for the dataset with multiple categorical attributes. DataWig makes an initial guess for the missing values using Mean imputation, and the prediction of missing values in one attribute needs to be based on the imputation results of other attributes. Thus, if it imputes one attribute incorrectly, it can greatly affect the accuracy of its imputation on other attributes. Baran achieves lower AMAE than DDGANI in two datasets, Spam and Thoracic, but it incorporates user supervision in the form of a number of manual imputations for examples with missing values, which leads to human costs. AimNet is also based on the attention mechanism and FDs for missing value imputation. But it only utilizes the attention scores between different attributes, while DDGANI not only considers the correlation between attributes but also different tuples, thereby fully utilizing the information provided from similar tuples to impute the missing values. Moreover, AimNet relies on the given FDs for missing value imputation without refinement, which can easily go wrong if the FDs are inaccurate. Graf utilizes CFDs for missing value imputation and co-cleans the inaccurate CFDs and missing values simultaneously. However, it employs sequence GANs to generate CFDs, which requires a lot of clean training data. And it can only impute the attribute values under the learned CFDs (In our experiments, we use Mean imputation for other attributes Graf could not touch). EGG-GAE works better with mixed datasets, and has lower ARMSE in the Australian dataset than DDGANI. It adopts graph neural networks for missing value imputation and reconstructs a tuple with missing value by leveraging neighboring nodes in the graph. Thus, it leads to difficulty in correctly imputing the missing values of "outlier", the tuple that can not find similar ones in the entire tabular data. Instead, DDGANI utilizes a generative model to rebuild the entire relational table step by step based on the distribution of all observed data, unlike EGG-GAE which strongly relies on similar tuples.

It is worth noting that all methods have high imputation errors in the Balance dataset. This is because most existing methods, including DDGANI, rely strongly on attribute correlations, making it difficult to impute the missing values appearing in the attribute that is not strongly correlated with other attributes. The correlations between different attributes are all zero in the Balance dataset, causing all methods to fail.

Table 4: Comparison with PLMs

Dataset	Metrics	ChatGPT	RPT	DDGANI
Wine	ARMSE	0.211	0.193	**0.145**
	AMAE	0.174	0.159	**0.106**
	Accuracy	94.44	97.22	**99.78**
Flare	ARMSE	0.258	0.250	**0.173**
	AMAE	0.258	0.250	**0.173**
	Accuracy	78.66	78.60	**80.37**
Adult	ARMSE	0.312	0.301	**0.195**
	AMAE	0.301	0.270	**0.180**
	Accuracy	82.13	83.33	**85.12**

Comparison with Pre-trained Language Models. Since the tuples in tabular data are often viewed as sequences [45], which is similar to a natural language processing task, we compare DDGANI additionally with pre-trained language models (PLMs) including ChatGPT and RPT [45] on the missing value imputation task. In this part of the experiments, we only show the results of Wine, Flare and Adult in Table 4 as the representatives of datasets with all numerical attributes, all categorical attributes and mixed types of attributes. Actually, our experiments on thirteen datasets all reveal that DDGANI has the best imputation performance, particularly on datasets consisting entirely of numerical data. The reason is that PLMs lack sufficient prior knowledge when confronted with purely numerical data. They are also less capable of capturing the underlying data distribution than deep generative models.

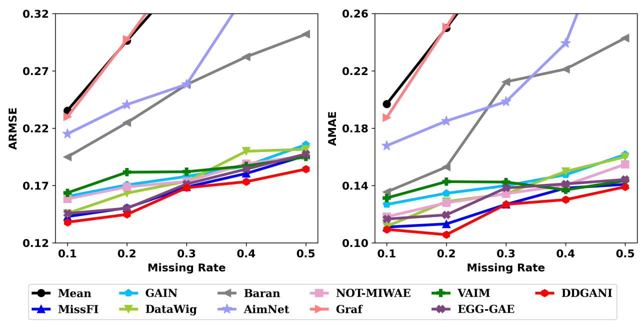
8.3 Comparison under Different Missing Rates

To better showcase the effectiveness of DDGANI, we vary the missing rate from 10% to 50% to test the performance of each method. In this set of experiments, we still choose Wine, Flare and Adult as the representatives of the datasets. The results are given in Figure 3.

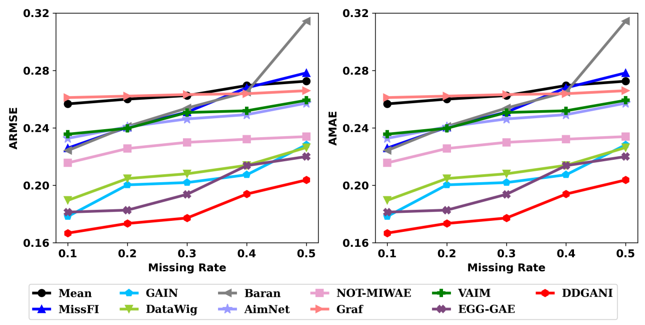
Obviously, as the missing rate increases, the performance of each imputation method declines. This is because the decrease in observed data makes it difficult to estimate the missing values, especially when the data near the missing values are gradually unavailable. Nevertheless, DDGANI consistently performs better than other baselines. As the missing rate increases, the imputation error of our method increases at a slower rate than that of GAIN and VAIM, which also employ generative models. Specifically, with the missing rate changing from 0.1 to 0.5 in Adult, the ARMSE of DDGANI increases from 0.179 to 0.247, while GAIN and VAIM increase from 0.213 to 0.298 and from 0.214 to 0.345, respectively. The reason behind this is that DDGANI adopts the diffusion model, which is better at predicting missing values from the context through several conditional denoising diffusion steps. Meanwhile, the usage of self-attention mechanisms and data dependency plug-in allows the model to better capture the correlation between missing values and the information of observed data.

8.4 Comparison under Different Missing Types

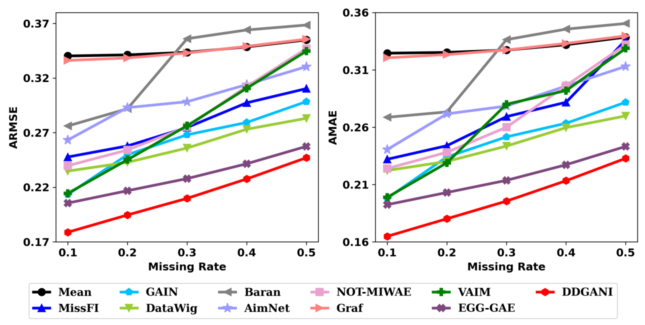
In this set of experiments, we evaluate the impact of different missing mechanisms on the performance of imputation methods. According to a recent study [21], we inject different types of missing values. 1) MCAR: removing some data from the dataset completely at random; 2) MAR: randomly selecting a numerical attribute to sort



(a) Wine

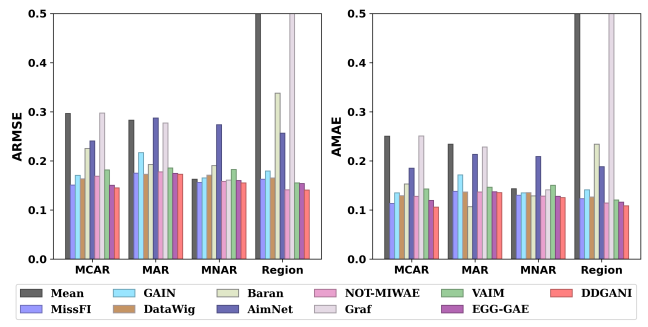


(b) Flare

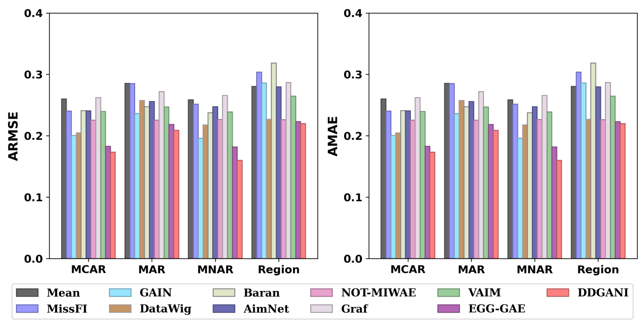


(c) Adult

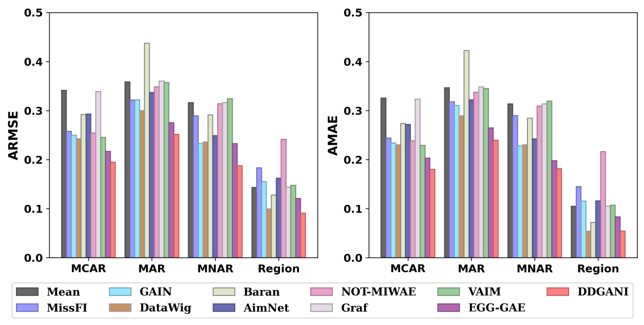
Figure 3: Comparison under Different Missing Rates



(a) Wine



(b) Flare



(c) Adult

Figure 4: Comparison under Different Missing Types

the tuples and then select some consecutive tuples to inject missing values into its other attributes; 3) MNAR: randomly removing two types of attribute values; one is the data in a numerical attribute that is less than the median, and the other is from a categorical attribute; Particularly, we inject 4) Regional Missing: designating a rectangular area by randomly selecting several consecutive attributes from some consecutive tuples and subsequently removing all data within this area. This kind of missing data often appears as the result of an outer join of relational tables.

The results shown in Figure 4 demonstrate the superiority of DDGANI. The advantages of our method are more evident under MAR, MNAR, and regional missing. Specifically, under the MAR mechanism, DDGANI achieves an ARMSE of 0.251 in Adult, notably superior to GAIN and VAIM with 0.322 and 0.357, respectively. This implies that our method can accurately capture the dependencies between attribute values. Regional missing leads to the loss of substantial sequential data. In this situation, our attention mechanism and data dependency plug-in help the model to place its eyes on more distant tuples even if it cannot find clues from nearby data to impute the missing values.

8.5 Utility of the Imputed Data

In this part of the experiments, we train a classifier with the imputed data and measure its accuracy on the test set to demonstrate the applicability of imputed training data. Here, we fix the missing rate at 20% under the MCAR mechanism and employ Random Forest as the downstream classifier. It is worth noting that Hospital and Tax are label-free. In order to conduct this experiment, we take the values in attribute EmergencyService and HasChild as labels for these two datasets, respectively, since these two attributes do not directly depend on other attributes but rather exhibit some correlations with most other attributes.

The experimental results are shown in Table 5, the first twelve columns. Intuitively, the smaller the imputation error, the higher

Table 5: Accuracy of Downstream Classifier

Dataset	Mean	MissFI	GAIN	DataWig	Baran	AimNet	NOT-MIWAE	Graf	VAIM	EGG-GAE	DDGANI	Removal	Original
Wine	94.44±0.00	99.44±1.11	97.78±1.11	98.89±1.36	94.44±1.76	93.52±1.31	98.89±1.36	95.00±3.24	98.89±1.36	98.89±1.36	**99.78±0.12**	88.89±0.00	100.00±0.00
Wireless	97.00±0.00	97.35±0.00	97.35±0.46	97.25±0.43	96.95±0.71	96.67±2.36	97.25±0.32	96.30±0.10	96.70±0.40	96.55±0.21	**97.70±0.25**	96.75±0.00	98.25±0.00
Spam	93.38±0.00	93.25±0.00	93.05±0.08	93.22±0.33	92.70±0.52	92.67±2.36	92.68±0.43	93.21±0.63	93.05±0.48	92.94±0.52	**93.55±0.31**	-	95.55±0.00
Tic-Tac-Toe	80.21±0.00	83.44±0.88	86.98±1.00	85.83±0.53	86.77±1.14	83.68±0.49	87.90±0.91	86.67±0.63	85.94±1.57	86.29±1.07	**88.72±1.30**	77.92±0.00	98.85±0.00
Flare	78.60±0.00	80.06±0.79	79.91±1.34	79.93±1.08	80.09±1.34	77.57±1.01	80.06±1.10	80.22±1.34	79.60±0.22	79.13±0.96	**80.37±1.10**	78.60±0.00	81.79±0.00
Balance	69.20±0.00	78.13±1.60	76.00±1.36	65.73±1.56	64.00±0.00	73.60±0.73	73.33±2.10	75.73±0.75	77.07±0.99	80.27±0.75	**80.33±0.31**	77.60±0.00	82.72±0.00
Adult	84.92±0.00	85.00±0.05	84.80±0.27	84.41±0.22	81.24±0.00	82.62±0.43	84.59±0.20	84.21±0.32	84.92±0.21	84.85±0.25	**85.12±0.09**	81.43±0.00	85.34±0.00
Hospital	96.50±0.00	96.50±0.00	96.60±0.20	95.80±0.87	96.70±0.24	96.83±0.24	96.60±0.20	96.70±0.24	96.50±0.0	96.70±0.27	**97.00±0.00**	95.00±0.00	97.00±0.00
Tax	86.48±0.00	87.04±0.82	85.75±0.56	86.30±0.42	84.93±0.72	84.91±0.37	86.82±0.40	86.77±0.61	86.58±0.61	86.43±0.27	**88.73±0.66**	86.29±0.00	89.41±0.00
Thoracic	78.72±0.00	80.14±1.00	79.57±0.43	78.72±0.00	78.94±1.04	78.01±1.33	78.72±0.95	78.09±0.85	79.15±0.52	78.72±0.21	**81.79±0.00**	77.45±0.00	82.36±0.00
Contraceptive	53.15±0.00	52.09±0.32	51.53±0.61	51.32±1.40	49.36±1.78	49.27±1.76	54.10±0.92	49.22±0.54	51.59±1.36	49.49±0.96	**55.59±0.36**	49.56±0.00	55.98±0.00
Australian	86.26±0.00	86.71±0.34	87.25±1.18	80.00±1.45	81.01±1.18	87.20±0.49	87.54±1.55	84.06±0.65	87.54±1.80	88.05±0.68	**88.13±0.29**	82.90±0.00	89.86±0.00
Beers	85.48±0.00	88.66±1.02	86.47±1.89	90.80±0.84	86.17±1.09	86.69±0.10	93.19±0.61	85.48±0.00	93.85±1.27	93.65±0.03	**96.26±0.54**	68.88±0.00	100.00±0.00

Table 6: An Ablation Study

Dataset	Metrics	DDGANI	*w/o* $\mathcal{L}_L$	*w/o* $\mathcal{L}_F$
Wine	ARMSE	0.145±0.001	0.157±0.002	-
	AMAE	0.106±0.001	0.121±0.002	-
	Accuracy	99.78±0.12	98.33±0.22	-
Flare	ARMSE	0.173±0.003	0.194±0.002	-
	AMAE	0.173±0.003	0.194±0.002	-
	Accuracy	80.37±1.10	78.88±1.16	-
Adult	ARMSE	0.195±0.000	0.206±0.000	0.201±0.002
	AMAE	0.180±0.000	0.192±0.000	0.190±0.001
	Accuracy	85.12±0.09	84.58±0.09	84.96±0.13
Tax	ARMSE	0.349±0.004	0.360±0.006	0.366±0.004
	AMAE	0.325±0.003	0.340±0.004	0.345±0.001
	Accuracy	88.73±0.66	86.70±0.30	87.62±0.62

Table 7: Comparison with Holistic (ARMSE)

Dataset	Rate	Holistic			DDGANI	
		T	P	R	P	R
Hospital	10%	**0.082**	0.123	0.103	0.114	0.091
	50%	0.396	0.513	0.456	0.401	**0.347**
Tax	10%	0.186	0.211	0.195	0.179	**0.164**
	50%	0.365	0.414	0.387	0.355	**0.327**

the accuracy of the downstream classifier. This has indeed been validated in many datasets. For example, in the Wine dataset, the Mean, AimNet, Baran, and Graf result in higher ARMSE, and the downstream classifier training by their imputed data is relatively poorer. However, as mentioned in [39], many factors, not just the imputation error of the training set, affect the performance of the downstream classifier. The impact of data quality on downstream classification tasks varies across different datasets. EGG-GAE, for example, does well in the missing value imputation task but cannot result in the superior performance of the downstream classifier. This is because of its difficulty in dealing with the outliers, which misleads the classifier. The deviation of the distribution of imputed outliers from the true distribution greatly affects the training of the classifier. DDGANI outperforms other methods in all datasets. The superior performance of DDGANI is attributed to the accurate imputation and the data utility plug-in, which could keep the imputed data as close to the original distribution as possible, allowing missing value imputation to be done in a direction that helps improve the performance of the downstream classifier.

Effectiveness of Data Utility Plug-in. We conduct an ablation study to delve deeper into the specific impact of data utility plug-in ($\mathcal{L}_L$) on the model performance. Table 6 shows that removing $\mathcal{L}_L$ does adversely affect the utility of imputed data on downstream tasks more than removing the data dependency plug-in ($\mathcal{L}_F$). To further verify that the optimization of $\mathcal{L}_L$ can improve data estimation without increasing bias, we also test the performance of the classifier training with the original dataset where no missing values are introduced (denoted by Original) and the dataset where tuples with missing values are removed directly (denoted by Removal). We can see from Table 5 that there is not much of a difference between DDGANI and Original except Tic-Tac-Toe. Particularly, we can achieve the same training results as the original Hospital dataset without missing values. Additionally, there is a noticeable improvement compared to the approach of not using any imputation algorithm (There are no complete tuples available in the Spam dataset for training the classifier after injecting 20% missing values).

8.6 Effectiveness of Data Dependency Plug-in

Comparison with Holistic. We inject 10% and 50% missing values into Hospital and Tax datasets under MCAR mechanisms to compare two ways of utilizing FDs for missing value imputation. We choose these two datasets since they contain more FDs. Since Holistic [14] directly utilizes the FDs to guide the imputation of missing values and cannot deal with the numerical attributes, we compare with it only regarding the attributes involved in FDs. Table 7 shows the ARMSE of imputation results. Here, "T", "P", and "R" denote the imputation of missing values with true FDs, potential FDs mined from missing data, and refined FDs, respectively. Table 7 shows that DDGANI with refined FDs performs better than Holistic employing true FDs in most cases. Moreover, the superiority of DDGANI becomes even more obvious as the missing rate increases. This may be attributed to DDGANI's capability not only to refer to the attributes involved in FDs but also to those outside the FDs when imputing the missing values. Additionally, it is noteworthy that both methods have better performance with refined FDs than initially obtained FDs, which reveals the role of FD refinement.

An Ablation Study. We also conduct an ablation study to verify the impact of data dependency plug-in ($\mathcal{L}_F$) on the model performance, which is shown in Table 6. Note that the removal of $\mathcal{L}_F$ has no impact on Wine and Flare since they have no FDs. But when dealing with a dataset like Tax that contains a large number of FDs, $\mathcal{L}_F$

significantly boosts model performance. We can see from Table 6 that the removal of $\mathcal{L}_F$ makes ARMSE increase from 0.349 to 0.366 and AMAE increase from 0.325 to 0.0.345 in Tax. It confirms the efficacy of $\mathcal{L}_F$ in preserving the data dependencies of imputed data.

Details of FDs Refinement. Here, we provide the details of the FDs refinement in the three datasets Adult, Hospital and Tax. Table 8 demonstrates the true FDs, the potential FDs mined from the tabular data with missing values, the refined FDs, and the number of refinement iterations needed. The red and gray FDs in Table 8 denote the erroneous FDs and the correct FDs missing after the FDs refinement, respectively.

First of all, we can notice that the number of potential FDs mined from the incomplete tabular data is not particularly large: the total number of true FDs in these three datasets is 19, while the number of candidate FDs mined is 30, which is almost 1.5 times the number of true FDs. The reason for this is we prune some of the candidates during the mining process and set a threshold to limit the maximum number of attributes probed on the left-hand side of an FD.

Table 8 further reveals that it is hard to avoid mining erroneous FDs from the tabular data with missing values, but the vast majority of them can be fixed correctly. We can see that we mine a total of 15 inaccurate FDs from the three datasets, of which 12 are correctly refined. Moreover, the refinement does not require too many iterations, which can prevent the erroneous FDs from having a continuous negative effect on the missing value imputation. However, there are three erroneous FDs that remain after the phase of refinement in Tax. This may be due to the strong correlation between the attributes in these FDs (*e.g.*, State→Areacode) or the values of attributes on the left-hand side being too discrete (*e.g.*, SingleExemp, MarriedExemp→Areacode). These three FDs hold true in most tuples, making it difficult to learn to impute the missing values that do not follow such patterns. It can also be seen that a correct FD in Hospital is mined in the beginning but mistakenly discarded in the refinement phase. The reason is that the imputation is not accurate enough, which makes the confidence of the tuple violating this FD slightly higher than the confidence of FD.

8.7 Impact of Hyperparameters

We vary the diffusion timestep T from 1 to 12 and the threshold κ in the attention mechanism from 0.05 to 1 to show their impact on the performance of missing value imputation. The results are shown in Figure 5.

As we can see from the three figures on the left in Figure 5, the imputation error decreases dramatically as T grows (The ARMSE decreases from 0.220 at $T = 1$ to 0.195 at $T = 6$ in Adult) but remains essentially stable when T exceeds a certain value, *e.g.*, $T = 6$. The results confirm the effectiveness of decomposing the generation process from a one-shot manner into multiple diffusion steps and the role of denoising diffusion GANs in shortening the diffusion process. As for the attention threshold κ, we observe a decline in imputation performance with increasing κ, which suggests that a smaller value of κ allows the model to focus more on the data associated with the missing values, effectively preventing decision biases caused by data imbalances. Due to the presence of the Learner, the accuracy of the downstream classifier varies within a certain controlled variance, less affected by these two hyperparameters.

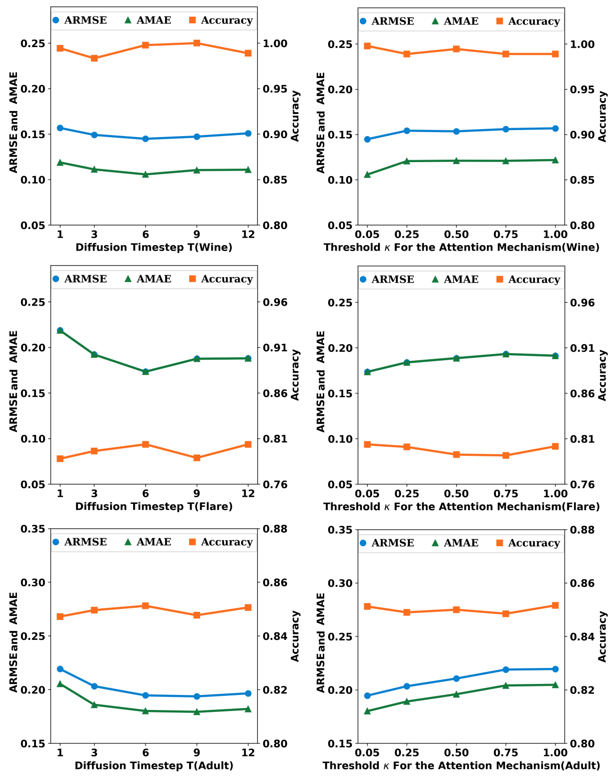


Figure 5: Impact of Hyperparameters

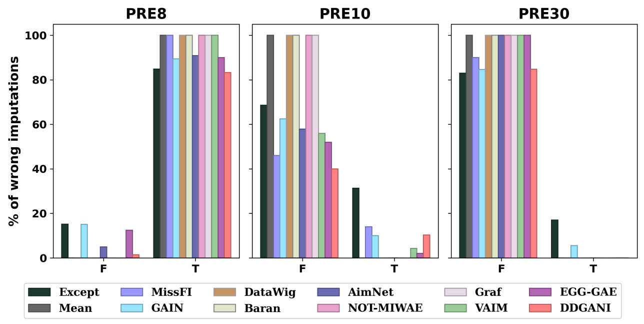


Figure 6: Capability to Handle Rare Values

8.8 Capability to Handle Rare Values

In order to evaluate the ability of DDGANI to deal with rare values, we have conducted an experiment on the Thoracic dataset based on the given work [9] with 20% missing rate under the MCAR model. The experimental results are shown in Figure 6. Here, the horizontal axis represents the attribute values, and there are only two attribute values “F” and “T” in the three attributes “PRE8”, “PRE10” and “PRE30”. The vertical axis represents the error rate of imputation for each method (For example, if ten cells need to be imputed with “F” but only one is imputed correctly, the error rate for “F” is 90%). The error rate of Except is $1 - f$ where f is the proportion of the corresponding attribute value in the complete ground-truth table. The higher the frequency of a value, the higher the probability it should be correctly imputed.

Table 8: Details of FDs Refinement

Dataset	True FDs	Potential FDs	Refined FDs	# Iterations
Adult	education-num→education education→education-num	education-num→education education→education-num marital-status, sex→relationship	education-num→education education→education-num	1
Hospital	HospitalName→City City→State HospitalName→State HospitalName→HospitalOwner MeasureCode→Condition MeasureName→Condition MeasureName→ MeasureCode MeasureCode→MeasureName	HospitalName→City MeasureCode→State MeasureName→State City→State HospitalName→State Condition, HospitalOwner→State HospitalName→HospitalOwner City, MeasureCode→HospitalOwner City, MeasureName→HospitalOwner MeasureCode→Condition MeasureName→Condition MeasureName→MeasureCode MeasureCode→MeasureName	HospitalName→City (missing) City→State HospitalName→State HospitalName→HospitalOwner MeasureCode→Condition MeasureName→Condition MeasureName→ MeasureCode MeasureCode→MeasureName	4
Tax	Areacode→State State, MaritalStatus→SingleExemp State, MarriedExemp→SingleExemp Areacode, MaritalStatus→SingleExemp Areacode, MarriedExemp→SingleExemp State, MaritalStatus→MarriedExemp State, SingleExemp→MarriedExemp Areacode, MaritalStatus→MarriedExemp Areacode, SingleExemp→MarriedExemp	State→Areacode SingleExemp, MarriedExemp→Areacode Areacode→State City, MarriedExemp→State SingleExemp, MarriedExemp→State Areacode→SingleExemp State→SingleExemp City, MaritalStatus→SingleExemp State, MaritalStatus→MarriedExemp State, SingleExemp→MarriedExemp Areacode, MaritalStatus→MarriedExemp Areacode, SingleExemp→MarriedExemp City, MaritalStatus→MarriedExemp City, SingleExemp→MarriedExemp	State→Areacode SingleExemp, MarriedExemp→Areacode Areacode→State SingleExemp, MarriedExemp→State Areacode, MaritalStatus→SingleExemp Areacode, MarriedExemp→SingleExemp State, MaritalStatus→SingleExemp State, MarriedExemp→SingleExemp State, MaritalStatus→MarriedExemp State, SingleExemp→MarriedExemp Areacode, MaritalStatus→MarriedExemp Areacode, SingleExemp→MarriedExemp	2

The results reveal that all methods, including DDGANI, have difficulty restoring to rare values. This issue likely arises since imputation models favor imputing missing cells with values occurring more frequently, which can lead to lower loss during the training phase. But DDGANI can achieve higher accuracy than other methods. This improved performance is likely attributed to applying the attention mechanism, which implements the action of selectively concentrating on the tuples with similar rare data in the same attribute for imputation. But if a value appears only once, there is really no chance for DDGANI to recover it, which is one limitation of our proposed method.

8.9 Runtime Analysis

In the end, we conduct experiments to compare the runtime of different methods and the results can be found in Figure 7. Since most of the baselines are based on DL models, the time we report is the runtime with a GPU. Without the GPU, many methods would take one to two days to achieve convergence.

We can see from Figure 7 that since we employ a more complex model to achieve a better imputation result, the running time of our method is not dominant. DDGANI significantly takes longer than other methods in the Hospital and Tax datasets. This is due to the need for the data dependency plugin to maintain and fix the dependencies between different attributes. But in other datasets, DDGANI does not need to take too much longer than other methods. For example, in the Spam, Tic-Tac-Toe, Adult, Thoracic, and Australian datasets, DDGANI's runtime is lower than that of DataWig, Baran, AimNet, Graf, and VAIM in many cases. This is because DDGANI adopts a diffusion model that approaches the target distribution step by step, with each step requiring only small distributional shifts. This makes the task relatively simple and stable, resulting in faster convergence than other methods.

Despite the slower runtime, DDGANI achieves better imputation results than other methods. Thus, DDGANI is better suited for scenarios where the accuracy of imputation, especially the accuracy of the downstream classifier, is desired.

9 CONCLUSION

In this paper, we have proposed DDGANI, a missing value imputation framework based on denoising diffusion GAN. We have incorporated the self-attention mechanism, which enables the generative model to capture more relevant contextual information. We have designed two plugins that can be chosen based on data characteristics and downstream tasks, which ensure the utility of imputed data and the preservation of inherent data dependencies. Extensive experimental results have demonstrated the approach outperforms state-of-the-art approaches.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (62372034, 61902017).

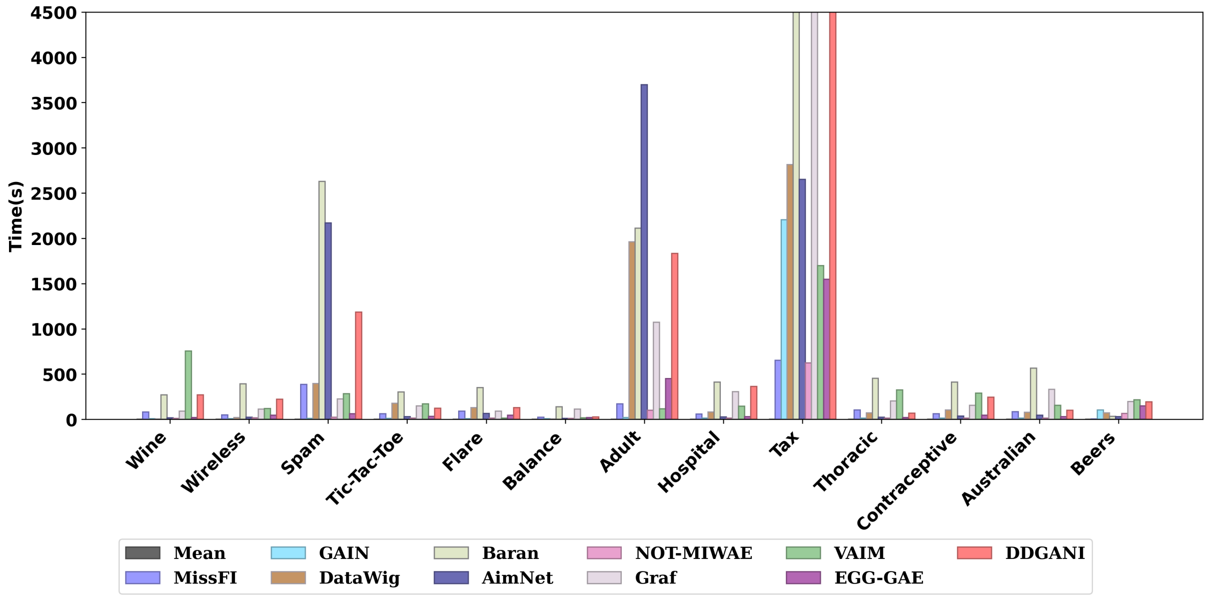


Figure 7: Comparison of Execution Time

REFERENCES

[1] Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician* 46, 3 (1992), 175–185.

[2] Anonymous Authors. 2023. Anonymous Title. IEEE.

[3] Olawale F Ayilara, Lisa Zhang, Tolulope T Sajobi, Richard Sawatzky, Eric Bohm, and Lisa M Lix. 2019. Impact of missing data on bias and precision when estimating change in patient-reported outcomes from a clinical registry. *Health and quality of life outcomes* 17 (2019), 1–9.

[4] Laure Berti-Équille, Hazar Harmouch, Felix Naumann, Noël Novelli, and Saravanan Thirumuruganathan. 2018. Discovery of genuine functional dependencies from relational data with missing values. *Proceedings of the VLDB Endowment* 11, 8 (2018), 880–892.

[5] Felix Biessmann, Tammo Rukat, Phillipp Schmidt, Prathik Naidu, Sebastian Schelter, Andrey Taptunov, Dustin Lange, and David Salinas. 2019. DataWig: Missing value imputation for tables. *Journal of Machine Learning Research* 20, 175 (2019), 1–6.

[6] Philip Bohannon, Wenfei Fan, Floris Geerts, Xibei Jia, and Anastasios Kementsietsidis. 2006. Conditional functional dependencies for data cleaning. In *2007 IEEE 23rd international conference on data engineering*. IEEE, 746–755.

[7] Bernardo Breve, Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. 2022. RENUVER: A Missing Value Imputation Algorithm based on Relaxed Functional Dependencies. In *EDBT*. 1–52.

[8] S. Van Buuren, J. P.L. Brand, C.G.M. Groothuis-Oudshoorn, and D. B.Rubin. 2006. Fully conditional specification in multivariate imputation. *Journal of Statistical Computation and Simulation* 76, 12 (2006), 1049–1064.

[9] Riccardo Cappuzzo, Saravanan Thirumuruganathan, and Paolo Papotti. 2024. Relational Data Imputation with Graph Neural Networks. In *EDBT/ICDT 2024, 27th International Conference on Extending Database Technology.*

[10] Loredana Caruccio, Vincenzo Deufemia, and Giuseppe Polese. 2015. Relaxed functional dependencies—a survey of approaches. *IEEE Transactions on Knowledge and Data Engineering* 28, 1 (2015), 147–165.

[11] Nadiia Chepurko, Ryan Marcus, Emanuel Zgraggen, Raul Castro Fernandez, Tim Kraska, and David Karger. 2020. ARDA: automatic relational data augmentation for machine learning. *Proceedings of the VLDB Endowment* 13, 9 (2020), 1373–1387.

[12] Fei Chiang and Renée J Miller. 2011. A unified model for data and constraint repair. In *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 446–457.

[13] Xu Chu, Ihab F Ilyas, and Paolo Papotti. 2013. Discovering denial constraints. *Proceedings of the VLDB Endowment* 6, 13 (2013), 1498–1509.

[14] Xu Chu, Ihab F Ilyas, and Paolo Papotti. 2013. Holistic data cleaning: Putting violations into context. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 458–469.

[15] Harald Cramér. 1999. *Mathematical methods of statistics*. Vol. 43. Princeton university press.

[16] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. Turl: Table understanding through representation learning. *ACM SIGMOD Record* 51, 1 (2022), 33–40.

[17] Alireza Farhangfar, Lukasz A Kurgan, and Witold Pedrycz. 2007. A novel framework for imputation of missing values in databases. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 37, 5 (2007), 692–709.

[18] Hanan Hammad Alharbi and Masaomi Kimura. 2020. Missing Data Imputation Using Data Generated By GAN. In *2020 the 3rd International Conference on Computing and Big Data*. 73–77.

[19] Xuerui Hong and Shuang Hao. 2023. Imputation of Missing Values in Training Data using Variational Autoencoder. In *2023 IEEE 39th International Conference on Data Engineering Workshops (ICDEW)*. IEEE, 49–54.

[20] Yka Huhtala, Juha Kärkkäinen, Pasi Porkka, and Hannu Toivonen. 1999. TANE: An efficient algorithm for discovering functional and approximate dependencies. *The computer journal* 42, 2 (1999), 100–111.

[21] Niels Bruun Ipsen, Pierre-Alexandre Mattei, and Jes Frellsen. 2021. not-MIWAE: Deep Generative Modelling with Missing not at Random Data. In *ICLR 2021-International Conference on Learning Representations*.

[22] Shawn R Jeffery, Minos Garofalakis, and Michael J Franklin. 2006. Adaptive cleaning for RFID data streams. In *Vldb*, Vol. 6. Citeseer, 163–174.

[23] Julie Josse, Jérôme Pagès, and François Husson. 2011. Multiple imputation in principal component analysis. *Advances in data analysis and classification* 5 (2011), 231–246.

[24] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[25] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. 2023. Tabddpm: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*. PMLR, 17564–17579.

[26] Steven Cheng-Xian Li, Bo Jiang, and Benjamin Marlin. 2018. MisGAN: Learning from Incomplete Data with Generative Adversarial Networks. In *International Conference on Learning Representations*.

[27] Ester Livshits, Alireza Heidari, Ihab F Ilyas, and Benny Kimelfeld. 2020. Approximate denial constraints. *arXiv preprint arXiv:2005.08540* (2020).

[28] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective error correction via a unified context representation and transfer learning. *Proceedings of the VLDB Endowment* 13, 12 (2020), 1948–1961.

[29] Mohammad Mahdavi, Ziawasch Abedjan, Raul Castro Fernandez, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2019. Raha: A configuration-free error detection system. In *Proceedings of the 2019 International Conference on Management of Data*. 865–882.

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

[31] John T McCoy, Steve Kroon, and Lidia Auret. 2018. Variational autoencoders for missing data imputation with application to a simulated milling circuit. *IFAC-PapersOnLine* 51, 21 (2018), 141–146.

[32] John T McCoy, Steve Kroon, and Lidia Auret. 2018. Variational autoencoders for missing data imputation with application to a simulated milling circuit. *IFAC-PapersOnLine* 51, 21 (2018), 141–146.

[33] Xiaoye Miao, Yangyang Wu, Lu Chen, Yunjun Gao, Jun Wang, and Jianwei Yin. 2021. Efficient and effective data imputation with influence functions. *Proceedings of the VLDB Endowment* 15, 3 (2021), 624–632.

[34] Robin Mitra, Sarah F McGough, Tapabrata Chakraborti, Chris Holmes, Ryan Copping, Niels Hagenbuch, Stefanie Biedermann, Jack Noonan, Brieuc Lehmann, Aditi Shenvi, et al. 2023. Learning from data with structured missingness. *Nature Machine Intelligence* 5, 1 (2023), 13–23.

[35] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. 2020. Handling incomplete heterogeneous data using vaes. *Pattern Recognition* 107 (2020), 107501.

[36] Jinfeng Peng, Derong Shen, Nan Tang, Tieying Liu, Yue Kou, Tiezheng Nie, Hang Cui, and Ge Yu. 2022. Self-supervised and Interpretable Data Cleaning with Sequence Generative Adversarial Networks. *Proceedings of the VLDB Endowment* 16, 3 (2022), 433–446.

[37] Theodoros Rekatsinas, Xu Chu, Ihab F Ilyas, and Christopher Ré. 2017. Holoclean: Holistic data repairs with probabilistic inference. *arXiv preprint arXiv:1702.00820* (2017).

[38] Patrick Royston and Ian R White. 2011. Multiple imputation by chained equations (MICE): implementation in Stata. *Journal of statistical software* 45 (2011), 1–20.

[39] Tolou Shadbahr, Michael Roberts, Jan Stanczuk, Julian Gilbey, Philip Teare, Sören Dittmer, Matthew Thorpe, Ramon Viñas Torné, Evis Sala, Pietro Lió, et al. 2023. The impact of imputation quality on machine learning classifiers for datasets with missing values. *Communications Medicine* 3, 1 (2023), 139.

[40] Shaoxu Song and Lei Chen. 2011. Differential dependencies: Reasoning and discovery. *ACM Transactions on Database Systems (TODS)* 36, 3 (2011), 1–41.

[41] Shaoxu Song, Yu Sun, Aoqian Zhang, Lei Chen, and Jianmin Wang. 2018. Enriching data imputation under similarity rule constraints. *IEEE transactions on knowledge and data engineering* 32, 2 (2018), 275–287.

[42] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. 2020. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks* 129 (2020), 249–260.

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

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

[45] Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Sam Madden, and Mourad Ouzzani. 2021. RPT: relational pre-trained transformer is almost all you need towards democratizing data preparation. *Proceedings of the VLDB Endowment* 14, 8 (2021), 1254–1261.

[46] Lev Telyatnikov and Simone Scardapane. 2023. EGG-GAE: scalable graph neural networks for tabular data imputation. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2661–2676.

[47] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. 2001. Missing value estimation methods for DNA microarrays. *Bioinformatics* 17, 6 (2001), 520–525.

[48] Stef van Buuren. 2007. Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical Methods in Medical Research* 16, 3 (2007), 219–242. https://doi.org/10.1177/0962280206074463

[49] Richard Wu, Aoqian Zhang, Ihab Ilyas, and Theodoros Rekatsinas. 2020. Attention-based learning for missing data imputation in HoloClean. *Proceedings of Machine Learning and Systems* 2 (2020), 307–325.

[50] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. 2021. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804* (2021).

[51] Jinsung Yoon, James Jordon, and Mihaela Schaar. 2018. Gain: Missing data imputation using generative adversarial nets. In *International Conference on Machine Learning*. PMLR, 5689–5698.