DIFFIMPUTE: TABULAR DATA IMPUTATION WITH DENOISING DIFFUSION PROBABILISTIC MODEL

Yizhu Wen[1], *Yiwei Wang*[2], *Kai Yi*[3], *Jing Ke*[4], *Yiqing Shen*[5,*]

[1]Spin-Quantum Co. Ltd, Xian, Shaanxi, China
[2]Z-ONE Technology Co. Ltd, Shanghai, China
[3]School of Mathematics and Statistics, University of New South Wales, Sydney, NSW, Australia
[4]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China
[5]Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA

ABSTRACT

Tabular data plays a crucial role in various domains but often suffers from missing values, thereby curtailing its potential utility. Traditional imputation techniques frequently yield suboptimal results and impose substantial computational burdens, leading to inaccuracies in subsequent modeling tasks. To address these challenges, we propose `DiffImpute`, a novel Denoising Diffusion Probabilistic Model (DDPM). Specifically, `DiffImpute` is trained on complete tabular datasets, ensuring that it can produce credible imputations for missing entries without undermining the authenticity of the existing data. Innovatively, it can be applied to various settings of Missing Completely At Random (MCAR) and Missing At Random (MAR). To effectively handle the tabular features in DDPM, we tailor four tabular denoising networks, spanning MLP, ResNet, Transformer, and U-Net. We also propose `Harmonization` to enhance coherence between observed and imputed data by infusing the data back and denoising them multiple times during the sampling stage. To enable efficient inference while maintaining imputation performance, we propose a refined non-Markovian sampling process that works along with `Harmonization`. Empirical evaluations on seven diverse datasets underscore the prowess of `DiffImpute`. Specifically, when paired with the Transformer as the denoising network, it consistently outperforms its competitors, boasting an average ranking of 1.7 and the most minimal standard deviation. In contrast, the next best method lags with a ranking of 2.8 and a standard deviation of 0.9. The code is available at `https://github.com/Dendiiiii/DiffImpute`.

Index Terms— Diffusion model, Tabular Imputation, Denoising networks, Harmonization

* Corresponding Author. Email: yshen92@jhu.edu

1. INTRODUCTION

Tabular data is crucial for data management and decision-making in various domains. However, missing values often compromise the utility of tabular data since most deep-learning methods require complete datasets, and the causes of it are inherently complex [1]. To counter this, researchers employ imputation methods, which are categorized into single and multiple imputation [2], to replace missing entries.

Single imputation, using techniques like mean and median imputation, can introduce bias by homogenizing missing entries with singular values, leading to a misrepresentation of the genuine data distribution [3]. On the opposite spectrum, multiple imputation suggests a gamut of plausible values for missing entries, leveraging iterative methods [4, 5] and deep generative models [6, 7]. Yet, these methods come with strings attached. Iterative methods might strain computational resources and demand robust data assumptions. Deep generative models, such as Generative Adversarial Networks (GANs) and Variation AutoEncoders (VAEs), grapple with challenges like mode collapse and posterior distribution alignment [8, 9]. Moreover, previous work [10] delineated an innovative score-centric approach, grounded on the gradient of the log-density score function. Therefore, the landscape still lacks a simple but efficient denoising diffusion stratagem crafted explicitly for tabular data imputation. In light of these challenges, we propose `DiffImpute`, a Denoising Diffusion Probabilistic Model (DDPM) specifically tailored for tabular data imputation. Unlike GANs and VAEs which are confined to Missing Completely At Random (MCAR) settings [11], the diffusion models can be applied to more generous settings like Missing At Random (MAR).

Drawing inspiration from the principles of image inpainting [12], our method first involves training the DDPM [13] on complete datasets. During inference, our method effectively replaces the missing entries within an observed dataset while preserving the integrity of the observed values. `DiffImpute` addresses mode collapse challenges observed in GAN-based approaches [14, 15] by the stability and simplicity of our training and inference process. Additionally, DiffImpute improves traceability by incorporating Gaussian noise throughout the diffusion process, as opposed to the prevalent practice of zero-padding in VAE-based approaches [16]. Correspondingly, we propose a novel Time Step Tokenizer to embed temporal order information into the denoising network. Based on this, we explore four different denoising network architectures, including MLP, ResNet, U-Net, and Transformer, to demonstrate the improvement of incorporating time information in the imputation process. Additionally, we propose Harmonization to meticulously aligns the imputed entries in data-deficient regions with the observed datasets through iterative processes of diffusion and denoising. Lastly, inspired by Song *et al*'s work [17], we introduces the Impute-DDIM to boost the imputation speed while keeping the imputation quality for the tabular data.

Our major contributions are four-fold: Firstly, we propose DiffImpute, a method that trains a diffusion model on complete data under MCAR and MAR missing mechanisms. DiffImpute offers a more stable and simplified training and inference process compared to other generative approaches. Secondly, DDPM, originally developed for image data, is adapted for tabular data by introducing the Time Step Tokenizer to encode temporal order information. This modification enables the customization of four tabular denoising network architectures: MLP, ResNet, Transformer, and U-Net in our experiment. Thirdly, we also introduce Harmonization to enhance coherence between imputed and observed data during the sampling stage. Finally, the Impute-DDIM allows repetitive and condensed time step sequences inference speed-up while implementing Harmonization. Correspondingly, we conduct extensive experiments on seven tabular datasets which suggest Transformer as the denoising network demonstrates faster training and inference, along with state-of-the-art performance.

2. METHODS

In this section, we elaborate on DiffImpute and unpack the four denoising network architectures correspondingly. Specifically, DiffImpute encompasses two stages: (1) the training of a diffusion model using complete tabular data; (2) the imputation of missing data from observed values.

2.1. Training Stage of DiffImpute.

The training phase of DiffImpute leverages DDPM on complete tabular data, denoted as $\mathbf{x}_0 = (x_0^1, x_0^2, \cdots, x_0^k) \in \mathbb{R}^k$, where k signifies the tabular data's dimensionality *i.e.,* the number of columns. Within DDPM, Gaussian noise $\boldsymbol{\epsilon}$ is introduced to drive the transition from input $\mathbf{x}_0$ to distorted latent feature $\mathbf{x}_t$ across a span of t time steps [13]. Then, the objective during the training of DiffImpute is to adeptly

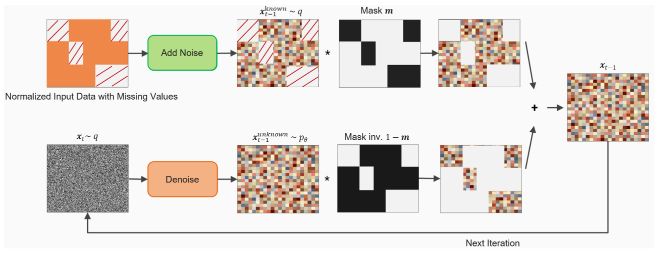
$\boldsymbol{x}_{t-1}^{known} \sim q$
Mask $\boldsymbol{m}$
Add Noise
Normalized Input Data with Missing Values
$\boldsymbol{x}_{t-1}$
$\boldsymbol{x}_t \sim q$
$\boldsymbol{x}_{t-1}^{unknown} \sim p_\theta$
Mask inv. $1 - \boldsymbol{m}$
Denoise
Next Iteration

Fig. 1: Schematic representation of DiffImpute. During inference, noisy data is extracted from known regions and supplemented with data imputed from the unknown region.

approximate the authentic data distribution of the complete tabular set. To accomplish this, a denoising network is trained to acutely predict the noise profile $\boldsymbol{\epsilon}$ that has been infused into $\mathbf{x}_t$. Specifically, we employ the smooth L1 loss function, motivated by the function's proficiency in discerning the discrepancies between the anticipated and the genuine noise [18]. Training pseudo-code is in the appendix.

Missing Data Imputation. In the sampling stage, the observed tabular data $\mathbf{x}$ is categorized into two distinct regions [12]. The "known region" defined by truly observed values is represented as $\mathbf{m} \odot \mathbf{x}$, where $\mathbf{m} \in \{0, 1\}^k$ is a Boolean mask pinpointing the known data with $\odot$ denoting element-wise multiplication. Conversely, the "unknown region" harbors the missing values, denoted by $(1 - \mathbf{m}) \odot \mathbf{x}$. Imputation is executed by leveraging our trained denoising network within DiffImpute, symbolized as $f_\theta(\mathbf{x}_t, t)$. This network focuses on the unknown region while retaining the values in the known sector, as illustrated in Fig. 1. Diving deeper, this denoising network embarks on a stepwise refinement of the "unknown region", commencing with unadulterated Gaussian noise. By tapping into the Markov Chain property of DDPM, Gaussian noise is injected at each time step t to aid in sampling from the known region, $\mathbf{m} \odot \mathbf{x}$, depicted as follows:

$$\mathbf{x}_{t-1}^{\text{known}} = \sqrt{\bar{\alpha}_{t-1}} \cdot \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1}} \cdot \boldsymbol{\epsilon}, \tag{1}$$

where $\bar{\alpha}_{t-1}$ signifies the aggregate diffusion level or noise imposed on the initial input data $\mathbf{x}_0$ until time step $t-1$, and $\boldsymbol{\epsilon} \in \mathbb{R}^k$ is drawn from a Gaussian distribution. However, for the unknown territories, the denoising network facilitates the sampling of progressively refined data with every backward step as follows:

$$\mathbf{x}_{t-1}^{\text{unknown}} = \frac{1}{\sqrt{\alpha_t}} \cdot \left(\mathbf{x_t} - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \cdot f_\theta(\mathbf{x}_t, t) \right) + \sigma_t \cdot \boldsymbol{\epsilon}, \tag{2}$$

where α_t represents the diffusion coefficient at time step t, σ_t denotes the posterior standard deviation at time step t. To synthesize the imputed data, the segments $\mathbf{x}_{t-1}^{\text{known}}$ and $\mathbf{x}_{t-1}^{\text{unknown}}$ are amalgamated based on their respective masks, yielding $\mathbf{x}_{t-1}$ at the $t-1$ time step:

$$\mathbf{x}_{t-1} = \mathbf{m} \odot \mathbf{x}_{t-1}^{\text{known}} + (1 - \mathbf{m}) \odot \mathbf{x}_{t-1}^{\text{unknown}}. \tag{3}$$

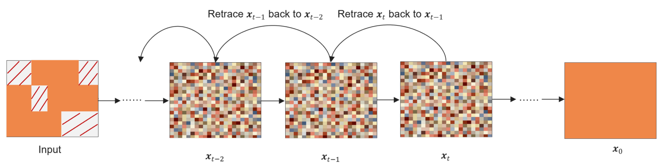
Retrace x_{t-1} back to x_{t-2}
Retrace x_t back to x_{t-1}
Input
x_{t-2}
x_{t-1}
x_t
x_0

Fig. 2: Harmonization process with t steps retrace back.

This procedure is reiterated in every reverse step until the final imputed data, x_0, emerges.

To further bolster the quality of our imputation, we propose Harmonization as a means to enhance the coherence between $\mathbf{x}_{t-1}^{\text{known}}$ and $\mathbf{x}_{t-1}^{\text{unknown}}$, thereby improving the quality of imputation. While Harmonization promises improved performance, extended time steps might inadvertently prolong inference runtime. To counterbalance this, we design Impute-DDIM to expedite the sampling process.

Harmonization. During the sampling of $\mathbf{x}_{t-1}$, we observed notable inconsistencies despite the model's active efforts to harmonize data at each interval [12], because the current methodologies are suboptimal in leveraging the generated components from the entire dataset. To overcome this challenge and enhance the consistency during the sampling stage, we introduce Harmonization (Fig. 2) to retrace the output $\mathbf{x}_{t-1}$ in Eq. (3) back by one or multiple steps to $\mathbf{x}_{t-1+j}$ by calculating $\sqrt{\alpha_t}\cdot\mathbf{x}_{t-1}+\sqrt{1-\alpha_t}\cdot\boldsymbol{\epsilon}$ incrementally j times, where $j \geq 1$ represents the number of steps retraced. For instance, $j = 1$ indicates a single-step retrace. It should be noted that as j increases, the semantic richness of the data is amplified. However, a trade-off emerges as the run-time during the inference phase grows since the denoising network having to initiate its operation from the time step $t-1+j$.

Impute-DDIM. To accelerate the sampling stage without compromising the benefits of Harmonization, we introduced Impute-DDIM, inspired by DDIM [17]. Central to its merit is the capacity to sample data at a substantially condensed time step τ for $\mathbf{x}_{t-1}^{\text{unknown}}$ during inference. By honing in on the forward procedure, specifically within the subset $\mathbf{x}_{\tau 1},\dots,\mathbf{x}_{\tau S}$ where $S \in \{1,\dots,T\}$, the computational weight tied to inference is appreciably reduced. Here, τ represents a sequentially increasing subset extracted from the range $\{1,\dots,T\}$. It's worth noting that the derivation of $\mathbf{x}_{t-1}^{\text{unknown}}$ from its preceding time step $\mathbf{x}_{t}^{\text{unknown}}$ underwent a slight alteration:

$$\mathbf{x}_{t-1}^{\text{unknown}} = \sqrt{\alpha_{t-1}}\cdot\left(\frac{\mathbf{x}_t-\sqrt{1-\alpha_t}f_\theta(\mathbf{x}_t^{\text{unknown}},t)}{\sqrt{\alpha_t}}\right)+ \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot f_\theta(\mathbf{x}_t^{\text{unknown}},t)+\sigma_t\boldsymbol{\epsilon}, \quad (4)$$

where $f_\theta(\mathbf{x}_t^{\text{unknown}},t)$ refers to the predicted noise at the time step for the unknown region of $\mathbf{x}$ using a trained denoising model. The sampling pseudo-code is in the appendix.

Overview. In brief, the overall sampling process of DiffImpute starts at time step T and backtracks to time step 0, the initial step involves drawing the noise-laden observation $\mathbf{x}_{t-1}^{\text{known}}$ at time step $t-1$. This is followed by its multiplication with the mask $\mathbf{m}$ to derive the known section. For the unknown region $(1-\mathbf{m})\odot\mathbf{x}$, $\mathbf{x}_{t-1}^{\text{unknown}}$ is sourced using the reverse procedure. The denoising network $f_\theta(\mathbf{x}_t,t)$ underpins this reverse modeling. Subsequently, the algorithm amalgamates the known and uncertain data facets to compute the imputed value at $t-1$. When the Harmonization setting with $j=1$ is active, a diffusion of the output $\mathbf{x}_{t-1}$ back to $\mathbf{x}_t$ is executed.

2.2. Denoising Network Architecture.

To obtain a denoising network tailored specifically for tabular data, we introduce the Time Step Tokenizer to encode temporal information into the denoising procedure. Building upon this foundational component, we have adapted four prominent denoising network architectures: MLP, ResNet, Transformer, and U-Net, as illustrated in Fig. 3.

Time Step Tokenizer. Time step tokenizer is designed to encapsulate the information of time step $t \in \mathbb{R}$, written as $\mathbf{t}_{\text{emb}} = \texttt{TimeStepTokenier}(t) \in \mathbb{R}^{2k}$. The tokenizer achieves this by formulating two distinct embeddings for scale and shift respectively, denoted as $\mathbf{t}_{\text{emb}} = \texttt{Concat}(\mathbf{t}_{\text{emb\_scale}},\mathbf{t}_{\text{emb\_shift}}) \in \mathbb{R}^{2k}$, where Concat signifies the concatenation of the two tensors $\mathbf{t}_{\text{emb\_scale}}$ and $\mathbf{t}_{\text{emb\_shift}}$ along the same dimension. These learnable embeddings, $\mathbf{t}_{\text{emb\_scale}}$ and $\mathbf{t}_{\text{emb\_shift}}$, are inspired by the fixed sine and cosine transformations of t [19], defined as:

$$\begin{aligned}\mathbf{t}_{\text{emb}} &= \texttt{Concat}(\mathbf{t}_{\text{emb\_scale}},\mathbf{t}_{\text{emb\_shift}})\\ &= \texttt{Linear}(\texttt{SiLU}(\texttt{Linear}(\texttt{GELU}(\texttt{Linear}\\ &\quad([\mathbf{t}_{\text{scale}},\mathbf{t}_{\text{shift}}]))))),\\ \mathbf{t}_{\text{scale}} &= \sin(t\cdot\exp\,(\frac{-\log(10^4)}{k}\cdot[0,1,\dots,k-1])),\\ \mathbf{t}_{\text{shift}} &= \cos(t\cdot\exp\,(\frac{-\log(10^4)}{k}\cdot[0,1,\dots,k-1])),\end{aligned}$$

where $\mathbf{t}_{\text{scale}} \wedge \mathbf{t}_{\text{shift}} \in \mathbb{R}^k$, Linear is a learnable linear layer, SiLU refers to the Sigmoid Linear Unit activation [20], and GeLU applies the Gaussian Error Linear Units function [21]. Thus, each of the $\mathbf{t}_{\text{emb\_scale}},\mathbf{t}_{\text{emb\_shift}}$ maintain the same dimension with $\mathbf{x}_t \in \mathbb{R}^k$. To integrate these time step embeddings with the feature $\mathbf{x}$, we compute the update as $\mathbf{x}\cdot(\mathbf{t}_{\text{emb\_scale}}+1)+\mathbf{t}_{\text{emb\_shift}}$, as depicted by "Add & Multiply" in Fig. 3(b).

MLP. By leveraging the time step tokenizer, we can adapt the MLP [22] to serve as a denoising network by incorporating t as an auxiliary input. Specifically, we introduce the time embedding, $\mathbf{t}_{\text{emb}}$, derived from the time step tokenizer, into a modified block named TimeStepMLP. This new block is an evolution of the traditional MLP Block. The architecture of this adaptation is depicted in Fig. 3(b) and can be mathemati-

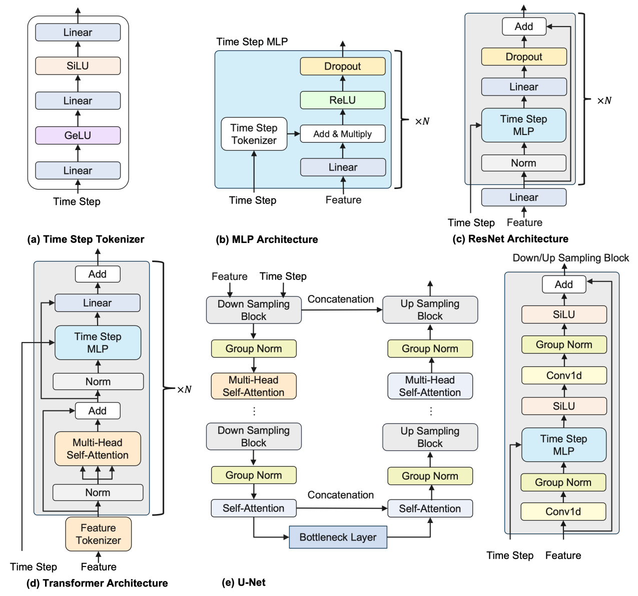

Fig. 3: Four types of denoising network architecture for tabular data. (a) Time Step Tokenzier, (b) MLP; (c) ResNet; (d) Transformer; (e) U-Net.

cally represented as

$$\texttt{MLP}(\mathbf{x}, \mathbf{t}_{\text{emb}}) = \texttt{Linear}(\texttt{TimeStepMLP}(\ldots(\texttt{TimeStepMLP}(\mathbf{x}, \mathbf{t}_{\text{emb}})))), \tag{5}$$

where detailed `TimeStepMLP` expression can be found in the appendix.

ResNet. Building on the foundation of the `TimeStepMLP`, we then introduce a variant of ResNet [22] tailored for tabular DDPM. In this design, the `TimeStepMLP` block is seamlessly integrated into each ResNet block, as illustrated in Fig. 3(c). We hypothesize that due to the depth of its representations, this ResNet variant will outperform the MLP-based models. Formally, the representation of our ResNet architecture is:

$$\texttt{ResNet}(\mathbf{x}, \mathbf{t}_{\text{emb}}) = \texttt{Prediction}(\texttt{ResBlock}(\ldots(\texttt{ResBlock}(\texttt{Linear}(\mathbf{x}), \mathbf{t}_{\text{emb}})))), \tag{6}$$

where detailed `ResBlock` expression can be found in the appendix.

Transformer. To further enhance our imputation capabilities, we adapt the Transformer architecture to tailor it explicitly for the tabular domain, as shown in Fig. 3(d). The transformer processes the feature and time step embeddings through a series of sequential layers, with each layer focusing on the feature level associated with a specific time stamp, t. To elevate the representation of input tabular data, $\mathbf{x}$, we employ a learnable linear layer, aptly named `Feature Tokenizer` [22]. Then, for a given feature $\mathbf{x} = (x^1, \cdots, x^k) \in \mathbb{R}^k$, its embeddings are constructed as $\mathbf{x}^k_{\text{emb}} = \mathbf{b}^k + x^k \cdot \mathbf{W}^k \in \mathbb{R}^d$, where $\mathbf{b}^k \in \mathbb{R}^d$ is the learnable bias and $\mathbf{W}^k \in \mathbb{R}^d$ represents the learnable weight. The aggregated embeddings are then represented as $\mathbf{x}_{\text{emb}} = [\mathbf{x}^1_{\text{emb}}, \ldots, \mathbf{x}^k_{\text{emb}}] \in \mathbb{R}^{k\times d}$, with d being the feature embedding dimension. To capture global contexts and further enhance the model's performance on downstream tasks, we introduce the $[\texttt{CLS}] \in \mathbb{R}^d$ token [23]. This token is concatenated with the embedding matrix $\mathbf{x}_{\text{emb}}$, resulting in $\texttt{Concat}([\texttt{CLS}], \mathbf{x}_{\text{emb}}) \in \mathbb{R}^{(k+1)\times d}$. The architecture can be mathematically described as:

$$\texttt{Transformer}(\mathbf{x}, \mathbf{t}_{\text{emb}}) = \texttt{Prediction}(\texttt{TransBlock}(\ldots(\texttt{TransBlock}(\texttt{Concat}([\texttt{CLS}], \texttt{FeatureTokenizer}(\mathbf{x})), \mathbf{t}_{\text{emb}})))) \tag{7}$$

where detailed `TransBlock` expression can be found in the appendix.

U-Net. U-Net [24] has garnered significant acclaim in the domain of diffusion models. Historically, its prowess has been predominantly demonstrated in image and text sequence processing. This has inadvertently led to a dearth of U-Net architectures specifically fine-tuned for tabular data. Addressing this gap, we introduce a novel U-Net tailored for tabular data, integrating both an encoder and decoder, as illustrated in Fig. 3(e). This design uniquely amalgamates a variant of `TimeStepMLP` and self-attention mechanisms, ensuring optimal performance for tabular data. Mathematically, our U-Net is represented as:

$$\texttt{UNet}(\mathbf{x}, \mathbf{t}_{\text{emb}}) = \texttt{Linear}(\texttt{DecoderBlock}(\cdots(\texttt{DecoderBlock}((\texttt{BottleneckBlock}(\cdots(\texttt{EncoderBlock}(\cdots \texttt{EncoderBlock}((\mathbf{x}, \mathbf{t}_{\text{emb}})))))))), \tag{8}$$

where detailed `DecoderBlock` and `EncoderBlock` expressions can be found in the appendix.

Denoising Network Formulation. Consequently, the denoising network is formulated as $f_\theta(\mathbf{x}, t) = \texttt{Network}(\mathbf{x}, \texttt{TimeTokenizer}(t))$. Here, `Network` can be any of the following architectures: `MLP`, `ResNet`, `Transformer`, or `U-Net`.

3. EXPERIMENTS

3.1. Dataset and Implementations.

Dataset. We leverage seven publicly accessible datasets, offering a diverse representation of domains. These datasets are: (1) California Housing (CA), real estate data [25]; (2) Helena (HE) and (3) Jannis (JA) are both anonymized datasets [26]; (4) Higgs (HI), simulated data of physical particles [27], where we adopted the version housing 98K samples from the OpenML repository [28]; (5) ALOI (AL), an image-centric dataset [29]; (6) Year (YE), dataset capturing audio

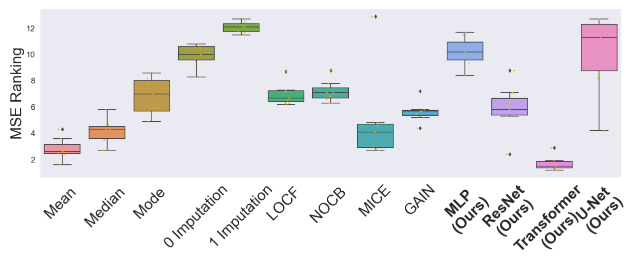

Fig. 4: Imputation performance rankings of imputation methods in terms of MSE. The lower the better.

features [30]; (7) Covertype (CO), it describes forest characteristics [31]. Preprocessing information is in the appendix.

Evaluation Metrics. To gauge the precision of imputed values, we manually induce random masks on the test set data. The randomness of the mask is characterized by a percentage $p_{\text{random}} \in \{10\%, \dots, 90\%\}$ for each row (MCAR) and column mask (MAR) number $p_{\text{col}} \in \{1, \dots, 4\}$. Three evaluative criteria have been established: (1) Mean Squared Error (MSE); (2) Pearson Correlation Coefficient; (3) Downstream Tasks Performance. Hyper-parameter information is in the appendix.

3.2. Results.

Comparison on Imputation Performance and Downstream Tasks. We start our evaluation by contrasting the performance of `DiffImpute` with a range of established single and iterative tabular imputation methods. As illustrated in Fig. 4 and Tab. 1, when equipped with a Transformer as the denoising network, `DiffImpute` consistently surpasses its peers, both in terms of MSE that measures the imputation performance and downstream tasks on the imputed data. However, an anomaly is observed with the HI dataset. Its second-place performance can be traced back to the dataset's distinct characteristics, notably its dominant normal distributions and scant tail densities. This particular outcome accentuates the effectiveness of the mean imputation technique. Interestingly, mean imputation not only holds its own but even outperforms well-regarded methods such as MICE, GAIN, and `DiffImpute` with ResNet. While MICE does outshine mean imputation in specific datasets like HE, AL, and YE, its overall rank suffers due to variable performance on other datasets. Within the sphere of deep generative models, GAIN's performance parallels that of `DiffImpute` with ResNet, albeit at a slower inference speed. Detailed experiment results are in the appendix.

Table 1: Downstream task rank performance comparison using the imputed dataset. For each dataset, ranks are calculated by averaging rank across all missing percentages of the metrics (RMSE for regression problem and accuracy score for classification problem) for each dataset. The lower the better.

Imputation Methods	**CA**	**HE**	**JA**	**HI**	**AL**	**YE**	**CO**	**Mean**	**Std**
Mean Imputation	3.9	4.5	6.5	**1.8**	6.9	3.9	4.3	4.5	1.7
Median Imputation	5.2	5.6	6.9	2.9	3.7	3.7	2.9	4.4	1.5
Mode Imputation	6.6	7.3	5.8	4.1	5.5	6.9	6.2	6.0	1.1
0 Imputation	10.1	9.2	8.1	7.6	7.9	8.0	9.5	8.7	1.0
1 Imputation	10.7	11.0	10.2	11.5	11.3	9.7	10.6	10.7	0.6
Last Observation Carrying Forward (LOCF)	8.2	10.5	10.1	9.7	11.5	10.5	8.5	9.9	1.2
Next Observation Carrying Backward (NOCB)	9.2	12.1	12.1	12.0	12.0	12.2	10.0	11.4	1.2
MICE	2.8	2.1	3.0	6.0	2.8	3.9	9.6	4.3	2.6
GAIN	4.9	3.5	4.0	7.3	4.9	5.2	7.7	5.4	1.6
`DiffImpute` w/ MLP	8.5	8.5	7.7	8.5	10.2	8.9	8.2	8.7	0.8
`DiffImpute` w/ ResNet	6.2	5.1	5.4	6.6	6.6	6.1	3.3	5.6	1.2
`DiffImpute` w/ Transformer	**1.5**	**2.2**	**2.4**	2.4	**1.4**	**3.4**	**1.4**	2.1	0.7
`DiffImpute` w/ U-Net	12.1	9.0	8.2	10.1	5.2	6.1	6.2	8.1	2.5

Effect of Denoising Network Architectures. Among the four denoising networks, the Transformer consistently stands out, marking its dominance in the tabular data domain. ResNets, on the other hand, serve as a robust baseline, delivering both impressive performance and swift inference speeds, thereby outperforming other models. The MLP and U-Net architectures face challenges in grasping sequential data, such as time step inputs. However, U-Net exhibits exceptional performance on the AL dataset, aligning with its foundational design for image data processing. Yet, its extended training and inference times make it a less optimal choice for tabular imputation. In summary, the Transformer within `DiffImpute` emerges as a leading solution.

Ablation Study. We conducted an ablation study on the time embedding layers, `Harmonization`, and `Impute-DDIM` on the CA dataset to gain insights into their individual contributions. We initiated our investigation by excluding the `time step tokenizer` from the denoising network. Interestingly, the impact on MSE performance was not uniform across models. This omission led to a noticeable decline in performance for the Transformer achitecture, with a 7.96% drop in MSE performance and 6.28% drop in the downstream task efficacy respectively. The U-Net and MLP architectures experienced significant improvements, recording a 63.81% and 94.76% enhancement in MSE, respectively. Subsequently, we evaluated the impact of incorporating the `Harmonization` with $j = 5$. The results, as detailed in Tab. 2, highlight the performance boosts achieved by `Harmonization` across various architectures. To illustrate, when integrated into the `DiffImpute` with the MLP model, there was a remarkable 53.81% increase in MSE and a 22.84% improvement in downstream task performance for the CA dataset. Lastly, we assessed the efficacy of `Impute-DDIM` in enhancing the inference speed, experimenting with different τ sampling steps, specifically $\tau \in \{10, 25, 50, 100, 250\}$ with $j = 5$. As shown in Tab. 3, when τ increases, the quality of imputation improves. Remarkably, with `Impute-DDIM` and a τ setting of 250, we managed to double the inference speed without compromising the MSE performance for both our MLP and Transformer models.

Table 2: Ablation on `Time Step Tokenizer` ('TST') and `Harmonization` ('H') with four denoising networks. We use the CA dataset and report the imputation performance in terms of MSE.

TST	H	MLP	ResNet	Transformer	U-Net
×	×	0.0212	0.0457	0.0210	0.0497
✓	×	0.0585	0.0498	0.0194	0.6831
×	✓	0.0164	0.0199	0.0174	0.0184
✓	✓	0.0268	0.0181	0.0191	4.2497

Table 3: Ablation on `Impute-DDIM` with four denoising networks. Note that when $\tau = 500$, no `Impute-DDIM` is applied.

τ	MLP	ResNet	Transformer	U-Net
10	0.2791	0.2574	0.2576	0.2741
25	0.2396	0.1892	0.1808	0.2274
50	0.1895	0.1164	0.0986	0.1727
100	0.1252	0.0525	0.0353	0.1145
250	0.0556	0.0240	0.0193	0.0795
500	0.0585	0.0498	0.0194	0.6831

4. CONCLUSION

In this work, we introduce `DiffImpute`, a novel denoising diffusion model for imputing missing tabular data. By seamlessly incorporating the `Time Step Tokenizer`, we have adapted four distinct denoising network architectures to enhance the capabilities of `DiffImpute`. Moreover, the amalgamation of the `Harmonization` technique and `Impute-DDIM` ensures that `DiffImpute` delivers superior performance without incurring extended sampling time. Our empirical evaluations, spanning seven diverse datasets, underscore the potential of `DiffImpute` as a foundational tool, poised to catalyze future innovations in the realm of tabular data imputation. One future direction is to further accelerate the sampling stage by distillation [32]. Additionally, we envision broadening the scope of `DiffImpute` to cater to missing multimodal scenarios, given that latent space features can be intuitively treated as tabular data.

5. REFERENCES

[1] Tan et al., "A tensor-based method for missing traffic data completion," *Trans Research: Emerging Tech*, vol. 28, pp. 15–27, 2013.

[2] Donald B. Rubin, *Multiple Imputation for Nonresponse in Surveys*, Wiley Series in Probability and Statistics. Wiley, 1987.

[3] Donald B. Rubin Roderick J. A. Little, *Statistical Analysis with Missing Data*, Wiley, 2002.

[4] Raghunathan et al., "A multivariate technique for multiply imputing missing values using a sequence of regression models," *Survey Methodology*, vol. 27, 11 2000.

[5] Van et al., "mice: Multivariate imputation by chained equations in r," *J. Stat. Softw.*, vol. 45, pp. 1–67, 2011.

[6] Lovedeep et al., "Mida: Multiple imputation using denoising autoencoders," 2018.

[7] Alfredo et al., "Handling incomplete heterogeneous data using vaes," 2020.

[8] Diederik et al., "An introduction to variational autoencoders," *FnTML*, vol. 12, no. 4, pp. 307–392, 2019.

[9] Ian et al., "Generative adversarial networks," 2014.

[10] Yidong et al., "Missdiff: Training diffusion models on tabular data with missing values," 2023.

[11] Daniel et al., "Hyperimpute: Generalized iterative imputation with automatic model selection," *ArXiv*, vol. abs/2206.07769, 2022.

[12] Lugmayr et al., "Repaint: Inpainting using denoising diffusion probabilistic models," 2022.

[13] Ho et al., "Denoising diffusion probabilistic models," 2020.

[14] Tim et al., "Improved techniques for training gans," 2016.

[15] Ian J. Goodfellow, "On distinguishability criteria for estimating generative models," 2015.

[16] Pierre-Alexandre et al., "Miwae: Deep generative modelling and imputation of incomplete data," 2019.

[17] Song et al., "Denoising diffusion implicit models," 2020.

[18] Kaan et al., "Generalized huber loss for robust learning and its efficient minimization for a robust statistics," 2021.

[19] Vaswani et al., "Attention is all you need," 2017.

[20] Stefan et al., "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," 2017.

[21] Dan et al., "Gaussian error linear units (gelus)," 2016.

[22] Gorishniy et al., "Revisiting deep learning models for tabular data," 2021.

[23] Jacob et al., "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[24] Ronneberger et al., "U-net: Convolutional networks for biomedical image segmentation," 2015.

[25] R. Kelley et al., "Sparse spatial autoregressions," *Stat. Probab. Lett.*, vol. 33, pp. 291–297, 1997.

[26] Guyon et al., "Analysis of the automl challenge series 2015–2018," *AutoML*, pp. 177–219, 2019.

[27] P. Baldi et al., "Searching for exotic particles in high-energy physics with deep learning," *Nat. Commun.*, vol. 5, pp. 291–297, 2014.

[28] Vanschoren et al., "Openml: Networked science in machine learning," *ACM SIGKDD Explor. Newslett.*, vol. 15, pp. 49–60, 12 2013.

[29] Geusebroek et al., "The amsterdam library of object images," *Int. J. Comput. Vis.*, vol. 61, pp. 103–112, 2005.

[30] Bertin-Mahieux et al., "The million song dataset.," *ISMIR*, pp. 591–596, 2011.

[31] Blackard et al., "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Comput. Electron. Agric.*, vol. 24, pp. 131–151, 1999.

[32] Tim et al., "Progressive distillation for fast sampling of diffusion models," 2022.