TabNAT: A Continuous-Discrete Joint Generative Framework for Tabular Data

Hengrui Zhang [*1] **Liancheng Fang** [*1] **Qitian Wu** [2] **Philip S. Yu** [1]

Abstract

While autoregressive models dominate natural language generation, their application to tabular data remains limited due to two challenges: 1) tabular data contains heterogeneous types, whereas autoregressive next-token (distribution) prediction is designed for discrete data, and 2) tabular data is column permutation-invariant, requiring flexible generation orders. Traditional autoregressive models, with their fixed generation order, struggle with tasks like missing data imputation, where the target and conditioning columns vary. To address these issues, we propose Diffusion-nested Non-autoregressive Transformer (TabNAT), a hybrid model combining diffusion processes and masked generative modeling. For continuous columns, TabNAT uses a diffusion model to parameterize their conditional distributions, while for discrete columns, it employs next-token prediction with KL divergence minimization. A masked Transformer with bi-directional attention enables order-agnostic generation, allowing it to learn the distribution of target columns conditioned on arbitrary observed columns. Extensive experiments on ten datasets with diverse properties demonstrate TabNAT's superiority in both unconditional tabular data generation and conditional missing data imputation tasks.

1. Introduction

The growing demand for synthetic tabular data in critical applications such as privacy-preserving data sharing and augmented machine learning pipelines (Fonseca & Bacao, 2023; Hernandez et al., 2022) has spurred significant interest in deep generative models. While multiple architectures, including VAEs (Liu et al., 2023), GANs (Xu et al., 2019), diffusion models (Zhang et al., 2024b), and LLMs (Borisov et al., 2023) have been explored, the potential of autoregressive modeling the dominant paradigm in language generation (Mann et al., 2020; Achiam et al., 2023)—remains underexplored for tabular data. This oversight persists despite tabular data's natural compatibility with sequential decomposition: $p(\mathbf{x}) = \prod_{i=1}^{D} p(\mathbf{x}^i|\mathbf{x}^{<i})$ where each $\mathbf{x}^i$ represents the i-th column of the table, and $\mathbf{x}^{<i}$ denotes the set of $\{\mathbf{x}^1, \cdots, \mathbf{x}^{i-1}\}$. We identify two fundamental limitations hindering the adoption of autoregressive models: 1) Type-specific modeling mismatch: Traditional autoregressive frameworks excel at handling discrete tokens through categorical distributions but struggle with continuous values unless using restrictive parametric assumptions or discretizations (Gulati & Roysdon, 2023). 2) Order-agnostic generation: Unlike language, which follows an inherent sequence, tabular columns exhibit permutation invariance[1]. Existing solutions that enforce fixed generation orders (Castellon et al., 2023) limit flexibility and introduce information loss. Furthermore, traditional autoregressive models that assume a fixed generation sequence struggle to adapt flexibly to tasks such as conditional sampling or missing data imputation, where the target columns and their conditioning context may vary.

On the other hand, Diffusion models (Ho et al., 2020; Song et al., 2021; Karras et al., 2022; Austin et al., 2021; Lou et al., 2024) are inherently well-suited for continuous data, as they iteratively denoise Gaussian noise to match the target distribution without imposing restrictive parametric assumptions. However, they struggle with discrete data, requiring inefficient and less effective adaptations such as embedding or categorical diffusion (Zhang et al., 2024b). Additionally, their parallel denoising process eliminates the need for a predefined generation order, naturally aligning with the permutation invariance of tabular data. While this parallelism enhances flexibility, it also limits the explicit modeling of column dependencies – a key strength of autoregressive approaches (see the comparison in Fig. 1).

To address these challenges, we propose Diffusion-nested **N**on-**A**utoregressive **T**ransformer (TabNAT), a novel framework that seamlessly integrates diffusion and autoregres-

[*]Equal contribution [1]Computer Science Department, University of Illinois at Chicago, Chicago, United States [2]Eric and Wendy Schmidt Center, Broad Institute of MIT and Harvard, Cambridge, United States. Correspondence to: Qitian Wu <wuqitian@broadinstitute.org>, Philip S. Yu <psyu@uic.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

[1]Column order carries no semantic meaning

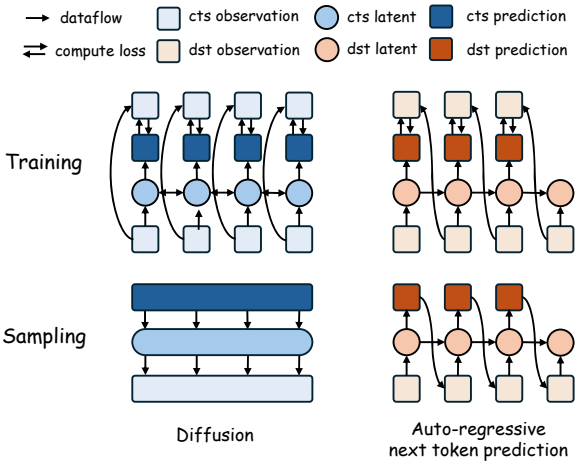


Figure 1: Comparison of Diffusion and Auto-regressive models: (Left) Diffusion models suited for continuous (cts) data with parallel generation capabilities, allowing simultaneous sampling across timesteps; (Right) An auto-regressive next-token prediction model designed for discrete (dst) data, requiring sequential generation one token at a time.

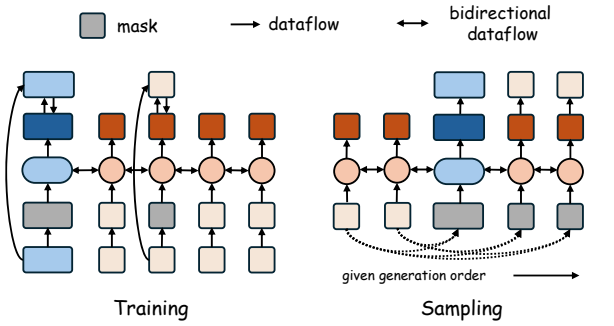


Figure 2: TabNAT supports both continuous and discrete variables through masked generative modeling. During training (left), the model learns to predict conditional distributions for masked positions in parallel. During sampling (right), the model generates values for masked positions sequentially following a given generation order. The bidirectional dataflow enables flexible conditioning on both past and future contexts, combining the benefits of parallel training with controlled sequential generation.

sive approaches for heterogeneous **Tab**ular data generation (Fig. 2). The core of TabNAT's dependency modeling lies in its masked generative architecture, where a bi-directional masked Transformer learns to generate conditional embeddings for target positions, which subsequently guide the modeling of their conditional distributions. For continuous columns, TabNAT employs a conditional diffusion model to capture their distributions based on the generated embeddings. For discrete columns, the model directly optimizes the categorical distributions by minimizing the KL divergence. This unified architecture enables TabNAT to handle mixed data types naturally and effectively. Furthermore, its non-autoregressive training paradigm, combined with flexible autoregressive sampling capabilities, facilitates both efficient parallel generation and order-aware inference, making it versatile for various inference tasks.

We conduct comprehensive experiments on ten tabular datasets of various data types and scales to verify the efficacy of the proposed TabNAT. Experimental results comprehensively demonstrate TabNAT 's superior performance in three key areas: **1) Statistical Fidelity**: The ability of synthetic data to faithfully recover the ground-truth data distribution; **2) Data Utility**: The effectiveness of synthetic data in downstream Machine Learning tasks, i.e., Machine Learning Efficiency and **3) Privacy Protection**: Ensuring that the synthetic data is sampled from the underlying distribution of the training data rather than being a direct replication. In missing data imputation tasks, TabNAT achieves remarkable performance, even surpassing state-of-the-art methods specifically designed for such tasks. We summarize the contributions of this paper as follows:

1) We propose TabNAT, a novel framework that integrates diffusion models with Bi-directional masked Transformers for heterogeneous tabular data generation.

2) We develop a unified architecture that seamlessly handles both continuous and discrete variables through conditional diffusion modeling and categorical distribution learning, respectively.

3) Extensive experiments on diverse real-world datasets demonstrate that TabNAT achieves state-of-the-art performance in both unconditional synthetic data generation and conditional missing data imputation tasks. Our ablation studies further validate the effectiveness of each component in our proposed framework.

2. Related Works

Synthetic Tabular Data Generation Generative models for tabular data have become increasingly important and have widespread applications (Assefa et al., 2021; Zheng & Charoenphakdee, 2022; Hernandez et al., 2022). For example, CTGAN and TAVE (Xu et al., 2019) deal with mixed-type tabular data generation using the basic GAN (Goodfellow et al., 2014) and VAE (Kingma & Welling, 2013) framework. DP-TBART (Castellon et al., 2023) and TabMT (Gulati & Roysdon, 2023) apply discretization techniques to numerical columns and then model the data distribution using Transformers (Vaswani et al., 2017), treating continuous features as discrete ones. Recently, inspired by the success of Diffusion models in image generation, a lot of diffusion-based methods have been proposed, such as

TabDDPM (Kotelnikov et al., 2023), STaSy (Kim et al., 2023), CoDi (Lee et al., 2023), and TabSyn (Zhang et al., 2024b), which have achieved SOTA synthesis quality. The proposed TabNAT combines Diffusion models and Transformers. It utilizes Transformers to generate conditional inputs for target columns, employs a conditional diffusion model to model continuous columns, and uses a categorical distribution model to handle discrete columns.

(Non-)Autoregressive Models for Continuous Data Autoregressive next-token generation is undoubtedly dominant in text generation (Mann et al., 2020; Achiam et al., 2023). However, in image generation, although autoregressive models (Van den Oord et al., 2016; Salimans et al., 2017) were proposed early on, their pixel-level characteristics limited their further development. Subsequently, diffusion models, which are naturally suited to modeling continuous distributions, have become the most popular method in the field of image generation. In recent years, some studies have attempted to use discrete-value image tokens (van den Oord et al., 2017; Razavi et al., 2019) and employ autoregressive transformers for image-generation tasks (Kolesnikov et al., 2022; Tian et al., 2024). However, discrete tokenizers are both difficult to train and inevitably cause information loss. To this end, recent work has attempted to combine continuous space diffusion models with autoregressive methods. For example, Li et al. (2024b) employs an autoregressive diffusion loss in a causal Transformer for learning image representations; Li et al. (2024a) proposes using a diffusion model to model the conditional distribution of the next continuous image and employs a masked bidirectional attention mechanism to enable the non-autoregressive generation of any number of tokens in arbitrary order.

3. Problem Formulation and Notations

In this paper, we always use uppercase boldface (e.g., $\mathbf{X}$) letters to represent matrices, lowercase boldface letters (e.g., $\mathbf{x}$) to represent vectors. Tabular data refers to data organized in a tabular format consisting of rows and columns. Each row represents an instance or observation, while each column represents a feature or variable. In this work, we consider heterogeneous tabular data that may contain both numerical and categorical columns or only one of these types. Let $\mathcal{D} = \{\mathbf{x}\}$ denote a tabular dataset, where each instance $\mathbf{x} = (x_1, x_2, \ldots, x_D)$ is a D-dimensional vector representing the values of D columns. We further categorize the columns into two types: 1) continuous (`cts`) columns; 2) discrete (`dst`) columns. Let $D_{\texttt{cts}}$ and $D_{\texttt{dst}}$ ($D_{\texttt{cts}} + D_{\texttt{dst}} = D$) be the number of continuous/discrete columns, respectively. Without loss of generality and for the sake of convenience, we assume that the first $D_{\texttt{cts}}$ columns of the table represent continuous variables, while the remaining $D_{\texttt{dst}}$ columns represent discrete variables. Then, correspondingly, we use $\mathbf{x}_{\texttt{cts}} \in \mathbb{R}^{D_{\texttt{cts}}}$ to denote the continuous columns of $\mathbf{x}$, $\mathbf{x}^i_{\texttt{dst}} \in \mathbb{N}$ to denote the i-th discrete column of $\mathbf{x}$, where $i = 1, \cdots, D_{\texttt{dst}}$, then $\mathbf{x} = [\mathbf{x}_{\texttt{cts}}, \mathbf{x}^1_{\texttt{dst}}, \cdots, \mathbf{x}^{D_{\texttt{dst}}}_{\texttt{dst}}]$.

4. TabNAT

4.1. Overview

In this section, we introduce the detailed architecture and implementations of TabNAT for modeling the distribution of heterogeneous tabular data.

Recall that our objective is to learn the joint distribution $p(\mathbf{x})$ via distribution decomposition, e.g.,

$$\begin{aligned} p(\mathbf{x}) &= p(\mathbf{x}_{\texttt{cts}}, \mathbf{x}^1_{\texttt{dst}}, \cdots, \mathbf{x}^{D_{\texttt{dst}}}_{\texttt{dst}}) \\ &= p(\mathbf{x}_{\texttt{cts}}) \prod_{i=1}^{D_{\texttt{cts}}} p(\mathbf{x}^i_{\texttt{dst}} | \mathbf{x}_{\texttt{cts}}, \mathbf{x}^{<i}_{\texttt{dst}}) \end{aligned} \tag{1}$$

and the corresponding log-likelihood is:

$$\log p(\mathbf{x}) = \log p(\mathbf{x}_{\texttt{cts}}) + \sum_{i=1}^{D_{\texttt{cts}}} \log p(\mathbf{x}^i_{\texttt{dst}} | \mathbf{x}_{\texttt{cts}}, \mathbf{x}^{<i}_{\texttt{dst}}). \tag{2}$$

Eq. 2 demonstrates that maximizing the likelihood of the data distribution can be achieved by maximizing the log-likelihood of each (conditional) distribution under the factorization. Given the permutation invariance property of tabular data, its joint distribution can be factorized along any arbitrary ordering. Consequently, this essentially requires us to learn every possible $p(\mathbf{x}_{\texttt{target}} | \mathbf{x}_{j \in \mathcal{S}})$, where $\mathcal{S}$ represents an arbitrary subset of the remaining columns. TabNAT employs a conditional generative model based on a bi-directional masked transformer to achieve the aforementioned objectives. Fig. 3 presents the overall framework of the proposed TabNAT, which takes a raw table of mixed continuous and discrete features as input and aims to model the distribution of masked positions conditioned on unmasked positions.

4.2. Model Architecture

Preprocessing and Embedding Layer. Raw input tables are typically unordered and irregularly structured (e.g., containing continuous features with varying scales and text-based discrete features). Therefore, in addition to the previously mentioned assumption that the first $D_{\texttt{cts}}$ columns are continuous features and the remaining $D_{\texttt{dst}}$ columns are discrete features, we first perform separate preprocessing steps for the continuous and discrete features, respectively.

For continuous columns, we standardize them to have zero mean and unit variance (i.e., a standard deviation of 1). For each discrete column i, we apply label encoding to transform

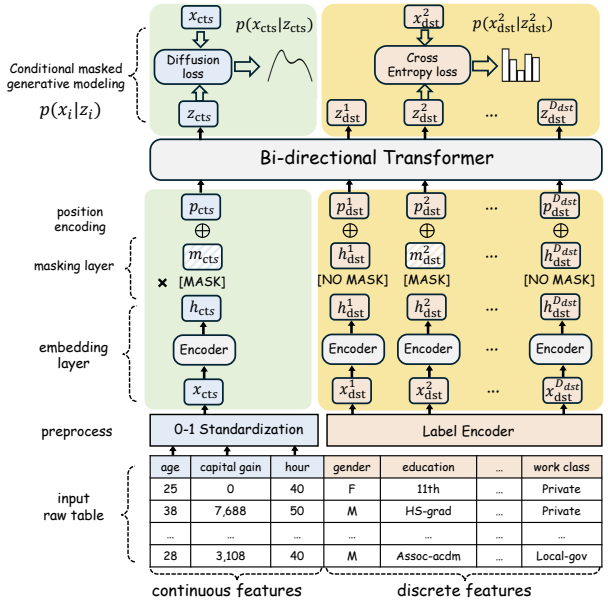


Figure 3: The overall framework of TabNAT. After preprocessing, an embedding layer first encodes each column into a vector. The masks are then added to the target columns ('age' and 'education'). With Bi-direction Transformers' decoding, the output vectors $\boldsymbol{z}$ are used as conditions for predicting the distribution of current columns. TabNAT uses the conditional diffusion loss for continuous columns and cross-entropy loss for discrete columns. Losses are computed only on the masked positions

them into index codes ranging from 0 to $|\mathcal{C}_i| - 1$, where $|\mathcal{C}_i|$ represents the number of possible categories of the i-th discrete feature. Thus, we can represent the continuous features as $\mathbf{x}_{\texttt{cts}} \in \mathbb{R}^{D_{\texttt{cts}}}$, and the i-th discrete feature as $\mathbf{x}^i_{\texttt{dst}} \in \{0, 1, \cdots, C_i - 1\}$.

Subsequently, we employ distinct continuous and discrete encoders to transform each type of variable into embeddings of identical dimensionality. For continuous columns, we utilize a simple two-layer MLP model ($\mathbb{R}^{D_{\texttt{cts}}} \to \mathbb{R}^d$) to map the $D_{\texttt{cts}}$-dimensional continuous features into a d-dimensional embedding. For the i-th discrete column, we allocate a d-dimensional (learnable) embedding vector to each distinct category. Finally, for each row of data $\mathbf{x}$ in the table, we obtain an embedding matrix as follows:

$$\mathbf{H} = [\mathbf{h}_{\texttt{cts}}, \mathbf{h}^1_{\texttt{dst}}, \cdots, \mathbf{h}^{D_{\texttt{dst}}}_{\texttt{dst}}] \in \mathbb{R}^{(1+D_{\texttt{dst}}) \times d}, \quad (3)$$

where $\mathbf{h}_{\texttt{cts}} \in \mathbb{R}^d$ is the embedding vector of $\mathbf{x}_{\texttt{cts}}$, and $\mathbf{h}^i_{\texttt{dst}}$ is the embedding vector of $\mathbf{x}^i_{\texttt{dst}}$, respectively. $\mathbf{H}$ as a sequence of length $1 + D_{\texttt{dst}}$ is subsequently taken as the input of a Bi-directional Masked Transformer.

Position masking. TabNAT utilizes mask embeddings to mark the target positions to predict. We first uniformly sample the number of positions to mask/predict $M \sim \mathcal{U}(1, 1 + D_{\texttt{dst}})$. Given M, we randomly sample a masking vector $\mathbf{m}_{\texttt{mask}} \in \{0, 1\}^D$, s.t., $\sum_i m^i_{\texttt{mask}} = M$. $m^i_{\texttt{mask}} = 1$ indicates that position i is masked and $= 0$ vice versa. Then, the embeddings of masked positions are replaced with position-specific mask embeddings, e.g.,

$$\mathbf{H}_{\texttt{mask}} = (1 - \mathbf{m}_{\texttt{mask}}) \odot \mathbf{H} + \mathbf{m}_{\texttt{mask}} \odot \mathbf{M}, \quad (4)$$

where $\mathbf{M} = [\mathbf{m}_{\texttt{cts}}, \mathbf{m}^1_{\texttt{dst}}, \cdots, \mathbf{m}^{D_{\texttt{dst}}}_{\texttt{dst}}]$ is the masking embedding matrix, and each $\mathbf{m}_{\texttt{cts}}$ or $\mathbf{m}^i_{\texttt{dst}}$ is a learnable mask embedding for the corresponding position.

Position encodings. Similar to conventional Transformer architectures, TabNAT incorporates position encoding for each individual position, i.e., $\mathbf{P} = [\mathbf{p}_{\texttt{cts}}, \mathbf{p}^1_{\texttt{dst}}, \cdots, \mathbf{p}^{D_{\texttt{dst}}}_{\texttt{dst}}] \in \mathbb{R}^{(1+D_{\texttt{dst}}) \times d}$. In our implementation, we employ learnable positional encodings that are continuously updated during training. This approach ensures permutation invariance for table columns, as these learnable positional encodings are column-specific and independent of the relative positions between different columns.

Bi-directional Transformer. With position encodings $\mathbf{P}$ added to the masked embeddings $\mathbf{H}_{\texttt{mask}}$, we employ a Bi-directional Transformer to capture the interrelationships among different positions in the sequence, which subsequently generates an output embedding matrix of identical dimensionality to the input. We let $\mathbf{Z} = [\mathbf{z}_{\texttt{cts}}, \mathbf{z}^1_{\texttt{dst}}, \cdots, \mathbf{z}^{D_{\texttt{dst}}}_{\texttt{dst}}] \in \mathbb{R}^{(1+D_{\texttt{dst}}) \times d}$ be the output of the last layer, where $\mathbf{z}_{\texttt{cts}}$ and $\mathbf{z}^i_{\texttt{dst}}$ are the output embeddings at corresponding positions.

4.3. Loss function and Training

Recall that we aim to learn $p(\mathbf{x}_{\texttt{target}}|\mathbf{x}_{j\in\mathcal{S}})$. Based on the aforementioned model design, $\mathbf{x}_{\texttt{target}}$ corresponds to a masked position, while $\mathbf{x}_{j\in\mathcal{S}}$ represents the unmasked positions. Given that each masked position's corresponding $\mathbf{z}_{\texttt{target}}$ is inherently a function of the unmasked inputs, TabNAT learns $p(\mathbf{x}_{\texttt{target}}|\mathbf{z}_{\texttt{target}})$ as a remedy for $p(\mathbf{x}_{\texttt{target}}|\mathbf{x}_{j\in\mathcal{S}})$.

Discrete columns. For a discrete column with input $\mathbf{x}^i_{\texttt{dst}}$ and output $\mathbf{z}^i_{\texttt{dst}}$, the target distribution follows a categorical distribution of dimension $|\mathcal{C}_i|$. Therefore, we employ a shallow MLP model ($\mathbb{R}^d \to \mathbb{R}^{|\mathcal{C}_i|}$) to map $\mathbf{z}^i_{\texttt{dst}}$ to a $|\mathcal{C}_i|$-dimensional space, followed by the application of the standard cross-entropy loss, i.e.,

$$\mathcal{L}^i_{\texttt{dst}} = \texttt{CE}(\texttt{MLP}^i_{\texttt{dst}}(\mathbf{z}^i_{\texttt{dst}}), \mathbf{x}^i_{\texttt{dst}}) \quad (5)$$

Continuous columns. For the continuous columns, TabNAT utilizes a conditional diffusion model (Karras et al., 2022; Song et al., 2021) to learn $p(\mathbf{x}_{\texttt{cts}}|\mathbf{z}_{\texttt{cts}})$, which has the following forward process:

$$\mathbf{x}^t_{\texttt{cts}} = \mathbf{x}^0_{\texttt{cts}} + \sigma(t)\boldsymbol{\varepsilon},\ \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \mathbf{x}^0_{\texttt{cts}} = \mathbf{x}_{\texttt{cts}}. \quad (6)$$

Algorithm 1 Loss for discrete columns

Input: Condition vector $\mathbf{z}^i_{\texttt{dst}}$, target discrete value $\mathbf{x}^i$, predictor $\texttt{MLP}^i_{\texttt{dst}}$
Output: Loss $\mathcal{L}^i_{\texttt{dst}}$
Get $\hat{\mathbf{x}}^i_{\texttt{dst}} = \texttt{MLP}^i_{\texttt{dst}}(\mathbf{z}^i)$
Compute loss: $\mathcal{L}^i_{\texttt{dst}} = \texttt{CE}(\hat{\mathbf{x}}^i_{\texttt{dst}}, \mathbf{x}^i_{\texttt{dst}})$

Algorithm 2 Loss for continuous columns

Input: Condition vector $\mathbf{z}_{\texttt{cts}} \in \mathbb{R}^d$, target continuous vector $\mathbf{x}_{\texttt{cts}} \in \mathbb{R}^{D^{\texttt{cts}}}$, x-encoder, z-encoder, denoising network $\texttt{MLP}_{\texttt{dn}}$
Output: $\mathcal{L}_{\texttt{cts}}$
Sample $t \sim p(t)$
Sample $\varepsilon \sim \mathcal{N}(0, 1)$
Get $\mathbf{x}^t_{\texttt{cts}} = \mathbf{x}_{\texttt{cts}} + \sigma(t) \cdot \varepsilon$
$\mathbf{x}^t_{\texttt{dn}} = x - \text{encoder}(\mathbf{x}^t_{\texttt{cts}})$
$\mathbf{z}_{\texttt{dn}} = z - \text{encoder}(\mathbf{z}_{\texttt{cts}})$
$\mathbf{t}_{\texttt{dn}} = \text{sinusoidal}(t)$
$\boldsymbol{\epsilon}_\theta(\mathbf{x}^t_{\texttt{cts}}, \mathbf{z}_{\texttt{cts}}, t) = \texttt{MLP}_{\texttt{dn}}(\mathbf{x}^t_{\texttt{dn}} + \mathbf{z}_{\texttt{dn}} + \mathbf{t}_{\texttt{dn}})$
Compute loss: $\mathcal{L}_{\texttt{cts}} = \|\boldsymbol{\epsilon}_\theta(\mathbf{x}^t_{\texttt{cts}}, \mathbf{z}_{\texttt{cts}}, t) - \varepsilon\|_2^2$

Then, the corresponding (conditional) reverse process is:

$$\begin{aligned} \mathrm{d}\mathbf{x}^t_{\texttt{cts}} = &- 2\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}^t_{\texttt{cts}}} \log p(\mathbf{x}^t_{\texttt{cts}}|\mathbf{z}_{\texttt{cts}})\mathrm{d}t \\ &+ \sqrt{2\dot{\sigma}(t)\sigma(t)}\mathrm{d}\boldsymbol{\omega}_t, \end{aligned} \tag{7}$$

where ω_t is the standard Wiener process. $\sigma(t)$ is the noise schedule, and $\dot{\sigma}(t)$ is the derivative of $\sigma(t)$ w.r.t. t. In this paper, we set $\sigma(t) = t$. Then, a sample can be obtained via integrating the reverse process from the maximum timestep $t = T$ until $t = 0$. We present the algorithms for computing the losses for discrete and continuous columns in Algorithm 1 and Algorithm 2, respectively.

A diffusion model is learned by using a denoising/score network $\boldsymbol{\epsilon}_\theta(\mathbf{x}^t_{\texttt{cts}}, \mathbf{z}_{\texttt{cts}}, t)$ to approximate the conditional score function $\nabla_{\mathbf{x}^t_{\texttt{cts}}} \log p(\mathbf{x}^t_{\texttt{cts}}|\mathbf{z}_{\texttt{cts}})$ (named score-matching). The final loss function could be reduced to a simple formulation where the denoising network is optimized to approximate the added noise $\boldsymbol{\varepsilon}$, i.e.,

$$\mathcal{L}_{\texttt{cts}} = \mathbb{E}_{t\sim p(t)}\mathbb{E}_{\boldsymbol{\varepsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\|\boldsymbol{\epsilon}_\theta(\mathbf{x}^t_{\texttt{cts}}, \mathbf{z}_{\texttt{cts}}, t) - \boldsymbol{\varepsilon}\|^2. \tag{8}$$

In Fig. 4, we illustrate the implementation of our denoising function. Given that $\mathbf{x}^t_{\texttt{cts}} \in \mathbb{R}^{D_{\texttt{cts}}}$, $\mathbf{z}_{\texttt{cts}} \in \mathbb{R}^d$, and $t \in \mathbb{R}$ possess different dimensionalities, we first project all of them into a shared space with dimension $d_{\texttt{diff}}$. For $\mathbf{x}^t_{\texttt{cts}}$, and $\mathbf{z}_{\texttt{cts}}$, we use x-encoder and z-encoder, which are essentially two learnable linear transformations, while for t, we utilize the $d_{\texttt{diff}}$-dimensional sinusoidal timestep embeddings. After that, they are added and serve as the input of a denoising MLP, which will subsequently output $\boldsymbol{\epsilon}_\theta(\mathbf{x}^t_{\texttt{cts}}, \mathbf{z}_{\texttt{cts}}, t)$. We present the entire training algorithm of TabNAT in Algorithm 3.

Algorithm 3 TabNAT: Training Process

Input: data $\mathbf{x} = (\mathbf{x}_{\texttt{cts}}, \mathbf{x}^1_{\texttt{dst}}, \cdots, \mathbf{x}^{D_{\texttt{dst}}}_{\texttt{dst}})$
Output: Model parameters
$\mathbf{h}_{\texttt{cts}} = \text{embedding}(\mathbf{x}_{cts})$
for $i \in 1, 2, \cdots, D_{\texttt{dst}}$ **do**
  $\mathbf{h}^i_{\texttt{dst}} = \text{embedding}(\mathbf{x}^i_{\texttt{dst}})$
end for
Let $\mathbf{H} = [\mathbf{h}_{\texttt{cts}}, \mathbf{h}^2_{\texttt{dst}}, \cdots, \mathbf{h}^{D_{\texttt{dst}}}_{\texttt{dst}}]$
Generate masking vector $\mathbf{m}_{\texttt{mask}}$
$\mathbf{H}_{\texttt{mask}} = (1 - \mathbf{m}_{\texttt{mask}}) \odot \mathbf{H} + \mathbf{m}_{\texttt{mask}} \odot \mathbf{M}$
$\mathbf{H}' = \mathbf{H}_{\texttt{mask}} + \mathbf{P}$
$\mathbf{Z} = \text{Transformers}(\mathbf{H}')$
Compute $\mathcal{L}_{\texttt{cts}}$ according to Algorithm 2
for $i \in 1, 2, \cdots, D_{\texttt{dst}}$ **do**
  Compute $\mathcal{L}^i_{\texttt{dst}}$ according to Algorithm 1
end for
$\mathcal{L} = \mathcal{L}_{\texttt{cts}} \cdot m_{\texttt{mask,cts}} + \sum_{i=1}^{D_{\texttt{dst}}} \mathcal{L}^i_{\texttt{dst}} \cdot m^i_{\texttt{mask,dst}}$
Back-propagation to optimize model parameters

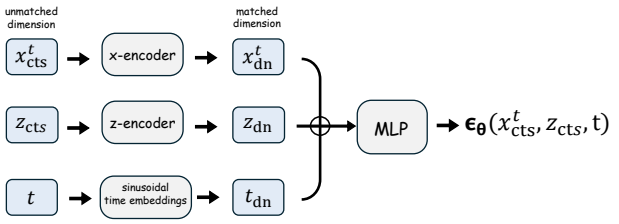


Figure 4: The architecture of the denoising function $\boldsymbol{\epsilon}_\theta(\mathbf{x}^t_{\texttt{cts}}, \mathbf{z}_{\texttt{cts}}, t)$. Since $\mathbf{x}^t_{\texttt{cts}}, \mathbf{z}_{\texttt{cts}}$, and t have distinct dimensions, they are first mapped to the same dimension, then added as the input of an MLP. The output of the MLP $\boldsymbol{\epsilon}_\theta(\mathbf{x}^t_{\texttt{cts}}, \mathbf{z}_{\texttt{cts}}, t)$ is utilized to predict the noise ε

4.4. Inference

After TabNAT is well-trained, we can easily perform a variety of inference tasks.

Unconditional data generation. To generate unconditional data examples $\mathbf{x} \sim p(\mathbf{x})$, we can first randomly sample a generation order, e.g. $\mathbf{x}_{\texttt{cts}} \to \mathbf{x}^1_{\texttt{dst}} \to, \cdots, \to \mathbf{x}^{D_{\texttt{dst}}}_{\texttt{dst}}$. Starting with all columns masked, we can first generate the values of all continuous columns, then generate the value of each discrete column one by one. It is worth emphasizing that TabNAT supports both traditional autoregressive next-token prediction and parallel generation of multiple tokens simultaneously, offering significant advantages in flexibility and efficiency.

Missing data imputation. Owing to TabNAT 's inherent capability to model various possible conditional distributions, it naturally supports conditional inference tasks, such as missing value imputation. The process is analogous to

Table 1: Comparison of different methods regarding the **Statistical Fidelity** and **Machine Learning Efficiency** of the synthetic data. All metrics have been normalized so that lower numbers indicate better performance, to facilitate better numerical comparison.

Method	Statistical Fidelity						MLE	
	Marginal↓ %	**Joint**↓ %	**α-Precision**↓ %	**β-Recall**↓ %	**C2ST**↓ %	**JSD**↓ 10^{-2}	**AUC Gap**↓ %	**RMSE Gap**↓ %
Interpolation								
SMOTE (Chawla et al., 2002)	1.72±1.36	2.95±1.66	3.78±3.94	**16.7**±9.16	2.26±0.52	0.11±0.10	3.00±3.66	0.15
VAE-based								
TVAE (Xu et al., 2019)	15.8±17.1	17.4±18.3	18.2±20.1	70.9±26.3	43.9±22.7	1.01±0.70	6.43±8.10	0.36
GOGGLE (Liu et al., 2023)	17.2±6.28	29.1±11.8	21.8±17.3	90.8±5.64	-	-	25.01±6.59	0.46
GAN-based								
CTGAN (Xu et al., 2019)	17.9±6.99	18.4±9.11	17.7±15.1	69.1±33.8	53.0±22.5	1.18±0.69	8.46±7.77	0.35
LLM-based								
GReaT (Borisov et al., 2023)	12.9±6.05	44.3±27.3	17.2±12.8	53.2±26.0	42.4±19.2	1.43±1.18	1.90±1.93	0.23
Diffusion-based								
STaSy (Kim et al., 2023)	14.3±7.40	13.5±9.76	21.8±24.7	55.6±29.6	53.9±16.6	1.25±1.13	2.34±2.24	0.17
CoDi (Lee et al., 2023)	17.4±11.3	15.2±19.8	10.0±5.93	51.7±31.1	44.0±33.3	0.76±0.50	7.08±12.15	0.24
TabDDPM (Kotelnikov et al., 2023)	15.0±25.3	7.92±8.16	23.6±2.93	49.6±34.5	24.6±38.9	1.03±1.60	6.80±16.71	2.23
TabSyn (Zhang et al., 2024b)	1.36±0.67	1.91±0.95	2.06±2.91	43.9±24.3	2.19±1.17	0.12±0.09	1.19±1.02	0.13
(Non)-Autoregressive-based								
DP-TBART (Castellon et al., 2023)	3.24±1.76	2.71±1.56	2.11±2.15	48.0±27.8	5.36±4.58	0.16±0.11	2.95±3.04	0.15
Tab-MT (Gulati & Roysdon, 2023)	14.9±15.1	8.11±10.8	23.2±31.9	63.5±38.3	48.6±47.7	0.60±0.83	3.88±4.55	0.92
TabNAT (ours)	**1.05**±0.39	**1.71**±0.72	**2.01**±1.10	48.4±20.1	**2.03**±1.42	**0.09**±0.048	**0.72**±0.9	**0.09**
Improvement	**22.8%**	**10.5%**	**4.7%**	-	**10.2%**	**18.2%**	**39.5%**	**30.8%**

unconditional generation, with the key difference being that it now only requires autoregressive generation at the missing positions in sequential order. It is noteworthy that when missing values are present in any continuous column, we mask the entire $\mathbf{x}_{\texttt{cts}}/\mathbf{h}_{\texttt{cts}}$ vector. With $\mathbf{z}_{\texttt{cts}}$, TabNAT utilizes the RePaint technique (Lugmayr et al., 2022; Zhang et al., 2024a), which iteratively refines the sampled values during the reverse denoising process based on the observed continuous columns. The detailed illustration is in Appendix B.5. We present the experiments of unconditional data generation and missing data imputation in Sec. 5 and Sec. 6, respectively.

5. Experiments: Synthetic Tabular Data Generation

5.1. Experimental Setups

Datasets. We select ten real-world tabular datasets of varying data types and sizes: 1) two containing only continuous features – *california* and *letter*; 2) two containing only categorical features – *car* and *nursery*; 3) six heterogeneous datasets of mixed continuous and discrete features – *adult*, *default*, *shoppers*, *magic*, *news*, and *beijing*. The detailed introduction of these datasets can be found in Appendix B.2.

Baselines. We compare the proposed TabNAT with ten powerful synthetic tabular data generation methods belonging to six categories. 1) The non-parametric interpolation method SMOTE (Chawla et al., 2002). 2) VAE-based methods TVAE (Xu et al., 2019) and GOGGLE (Liu et al., 2023). 3) GAN-based method CTGAN (Xu et al., 2019). 4) LLM-based method GReaT (Borisov et al., 2023). 5) Diffusion-based methods: STaSy (Kim et al., 2023), CoDi (Lee et al., 2023), TabDDPM (Kotelnikov et al., 2023), and TabSyn (Zhang et al., 2024b). 6) Autoregressive methods DP-TBART (Castellon et al., 2023) and Tab-MT (Gulati & Roysdon, 2023). The proposed TabNAT is close to both Diffusion and autoregressive methods, since it nests a diffusion model into a (non)-autoregressive framework in order to handle continuous features.

Evaluation. Following previous works (Zhang et al., 2024b), we evaluate the quality of synthetic tabular data across three key dimensions: 1) **Statistical Fidelity**: Whether the synthetic data accurately recovers the underlying distribution of the ground-truth data. 2) **Utility**: The performance of synthetic data in downstream tasks, measured by Machine Learning Efficiency (MLE). 3) **Privacy Protection**: Whether the synthetic data avoids copying real records, ensuring data privacy. Detailed descriptions of the evaluation metrics can be found in Appendix B.7. We each model and dataset, we sample 20 synthetic dataset of the same size as the training set. More implementation details are provided in Appendix B.4.

5.2. Main Results

Statistical Fidelity. We first investigate whether synthetic data can faithfully reproduce the distribution of the original data. We use various metrics from different aspects, including *Marginal*, *Joint*, *α-Precision*, *β-Recall*, *C2ST*, and *JSD*. Due to space limitations, in this section, we only present the average performance with standard deviation on each metric across all ten datasets. The detailed performance on each individual dataset is in Appendix C.1.

In Table 1, we present the performance comparison on these

Table 2: Impacts of the key components of TabNAT .

Variants	Marginal	Joint
with MSE loss	12.31%	18.27%
with Gaussian	5.44%	7.59%
with fixed order	1.86%	2.74%
TabNAT	**1.05%**	**1.71%**

Table 3: Impacts of the model depth (number of Transformer layers) and width (embedding dimension d) on the fidelity of synthetic data.

Depth	Marginal	Joint	Dim	Marginal	Joint
2	1.79%	2.85%	8	2.88%	3.79%
4	1.44%	2.37%	16	1.28%	2.09%
6	**1.05%**	**1.71%**	32	**1.05%**	**1.71%**
8	1.19%	2.13%	64	1.22%	1.93%

fidelity metrics. As demonstrated, TabNAT achieves performance far surpassing the second-best method in five out of six fidelity metrics, with advantages ranging from 10.5% to 39.5%. Considering that these metrics have already reached a considerably high level due to the rapid development of recent deep generative models for tabular data, our improvement is substantial. The only exception is β-Recall, on which the simple interpolation method SMOTE achieves the best performance. This is because they perform linear or nonlinear interpolation directly between existing real samples, so the generated samples will definitely fall within the support set of the real data distribution, making them more likely to have similar feature distributions to the real minority class samples. However, the limitation of interpolation methods lies in the fact that the synthetic data is too close to the real data, making it resemble a copy from the training set rather than a sample from the underlying distribution, which may cause privacy issues. Detailed experiments are in the Privacy Protection section.

Utility on Downstream Tasks. We then evaluate the quality of synthetic data by assessing its performance in the Machine Learning Efficiency (MLE) task. The performance difference is measured by the difference between AUC/RMSE obtained from synthetic data and AUC/RMSE obtained from real training data. As demonstrated in Table 1, the proposed TabNAT achieves satisfactory performance on MLE tasks, yielding over 30% improvement over the most competitive baselines. These results demonstrate that the synthetic data generated by TabNAT is comparable to real training data for training machine learning classification/regression models.

Privacy Protection. Finally, we use the Distance to Closest Record (DCR) score for measuring the capacity of TabNAT in privacy protection. In Table 5 and Figure 5, we present the numerical comparison and the visualizations of the DCR distributions of SMOTE, TabDDPM, TabSyn, DP-TBART, and the proposed TabNAT. As demonstrated, SMOTE performs poorly in privacy protection, as its synthetic sample tends to be closer to the training set rather than the holdout set, since its synthetic examples are obtained via interpolation between training examples. By contrast, the remaining deep generative methods are all effective at preserving the privacy of the training data, resulting in almost completely overlapped DCR distributions between the training set and the holdout set.

Table 4: Comparison of training, sampling time, and fidelity metrics of different methods on Adult dataset.

Method	Training	Sampling	Marginal	Joint
CTGAN	1030s	0.8621s	17.9%	18.4%
TVAE	353s	0.5118s	15.8%	17.4%
GReaT	2h 27min	2min 19s	12.9%	44.3%
STaSy	2283s	8.941s	14.3%	13.5%
CoDi	2h 56min	4.616s	17.4%	15.2%
TabDDPM	1031s	28.92s	15.0%	7.92%
TabSyn	2422s	11.84s	1.36%	1.91%
DP-TBART	1855s	2.11s	3.24%	2.71%
Tab-MT	1882s	2.18s	14.9%	8.11%
TabNAT	2754s	15.69s	**1.05%**	**1.71%**

5.3. Ablation Studies

Effects of the key components of TabNAT . We first study if TabNAT benefits from 1) the diffusion loss and 2) arbitrary generation order in the synthetic data generation task. To validate the importance of the diffusion loss, we consider two alternative approaches: (1) directly predicting the values of continuous columns using an MSE loss, and (2) parameterizing the conditional probability of each continuous column as a Gaussian distribution, with the model predicting its parameters. To investigate the benefits of arbitrary generation order, we employed fixed causal masking during training based on the default column ordering of the table and generated synthetic tables following this predefined sequence. In Table 2, we compare TabNAT with the three variants on the average synthetic data's fidelity w.r.t. *Marginal* and *Joint* across all the datasets. We can observe that both the diffusion loss and random training/generation order are important to TabNAT 's success. Specifically, the diffusion loss targeting the modeling of the conditional distribution of the continuous columns contributes the most, and the random ordering further improves TabNAT 's performance. Furthermore, random ordering enables TabNAT to perform flexible conditional inference tasks like imputation.

Sensitivity to hyperparameters. We then study TabNAT 's sensitivity to its hyperparameters that specify its transformer architecture: the model depth (number of Transformer layers) and the embedding dimension d. From Table 3, we can observe that although there is an optimal hyperparameter setting (i.e., depth = 6, d = 32), changes in these two hyperparameters do not significantly harm the model's performance. These results demonstrate that our model is relatively robust

Table 5: Probability that a synthetic example's DCR to the training set is smaller than that of the holdout set, a value closer to 50% is better.

Method	Default	Shoppers
SMOTE	91.41%	96.40%
TabDDPM	51.30%	51.74%
TabSyn	50.88%	51.50%
DP-TBART	50.83%	51.27%
TabNAT	51.13%	50.97%

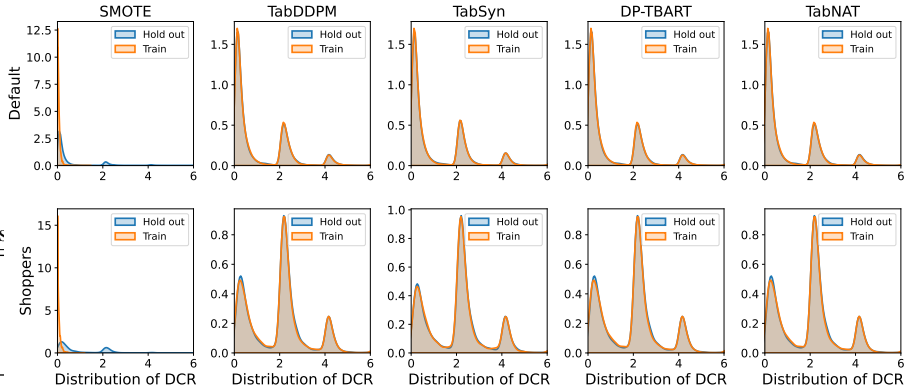


Figure 5: Distribution of DCR scores from training and holdout set

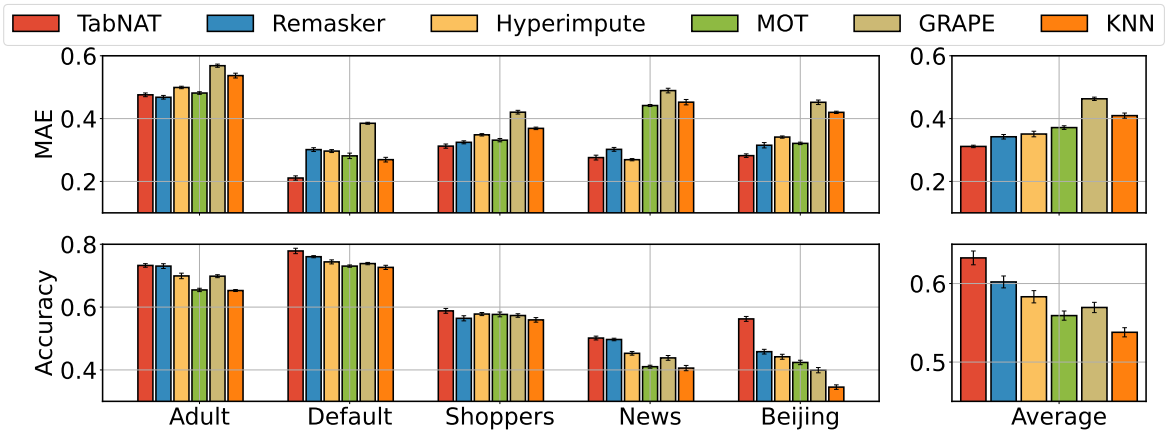


Figure 6: (30% MCAR) Comparison of missing value imputation performance with SOTA imputation methods on five heterogeneous datasets. Top: Average MAE on continuous features. Bottom: Average Accuracy on discrete features.

to these hyperparameters. Therefore, good results can be obtained for different datasets without the need for specific hyperparameter tuning.

Comparison of training/sampling time. In Table 4, we compare the training and sampling time of TabNAT with other methods on the Adult dataset, which contains 6 continuous columns, 8 discrete columns and 22, 792 samples.

As shown in the table results, despite employing a dual architecture combining diffusion and autoregressive modeling, TabNAT 's training speed does not significantly lag behind other competitive models such as TabSyn and DP-TBART. In terms of sampling efficiency, since the primary computational overhead lies in the diffusion process for continuous features, and TabNAT requires only a single diffusion pass to sample all continuous columns, its sampling speed is comparable to TabSyn. Considering the substantial improvement in synthetic data quality achieved by TabNAT, this trade-off is highly justified.

6. Experiments: Missing Data Imputation

6.1. Experimental Setups

Datasets. We select five datasets in Sec. 5.1 that contain heterogeneous data types: *Adult*, *Default*, *Shoppers*, *News*, and *Beijing*. Since TabNAT is not primarily designed for this task, we only consider an out-of-sample imputation setting, where the complete training data is observable, and we aim to predict the missing entries in the testing set. The missing data mechanism follows Missing Completely at Random (MCAR), with a missing rate of 30% applied to the test set. Specifically, each column has an independent probability of 0.3 of containing missing values.

Baselines. We compare the proposed TabNAT with the state-of-the-art (SOTA) methods for missing data imputation, including KNN (Pujianto et al., 2019), GRAPE (You et al., 2020), MOT (Muzellec et al., 2020), HyperImpute (Jarrett et al., 2022) and Remakser (Du et al., 2024).

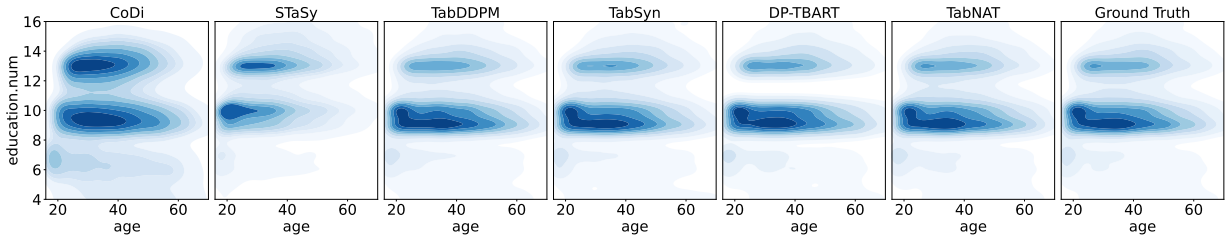


Figure 7: 2D joint distribution density of 'age' and 'education num' of Adult dataset.

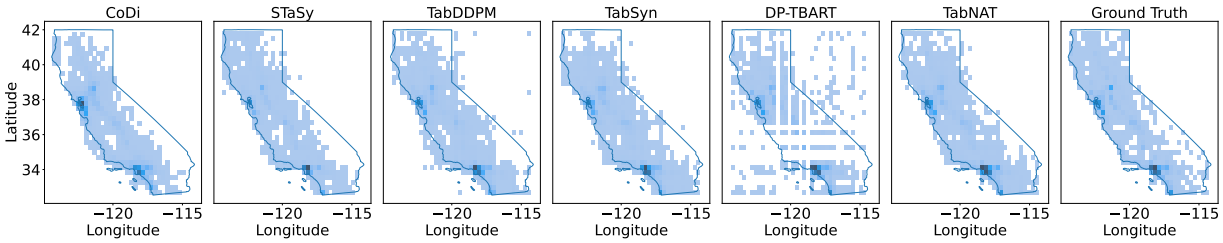


Figure 8: 2D joint distribution density of 'Longitude' and 'Latitude' of California dataset.

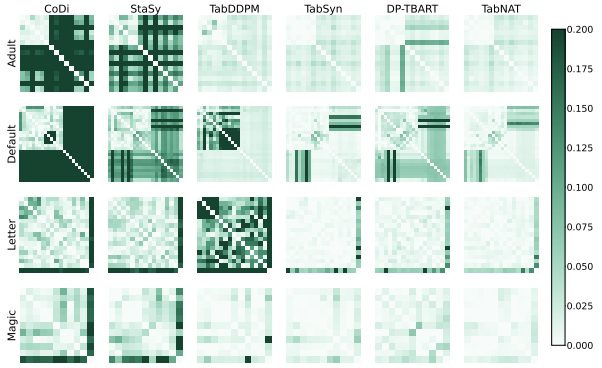


Figure 9: Heatmaps of the joint column correlation estimation error. Lighter areas indicate a lower error in estimating the correlation between two columns.

6.2. Main Results

In Figure 6, we present the performance comparison on each of the five datasets as well as the average imputation performance. Since these datasets contain both continuous and discrete features, we employ the Mean Absolute Error (MAE) metric to evaluate imputed continuous features, while using accuracy as the evaluation metric for discrete features. As demonstrated, the proposed model exhibits superior performance across all five datasets, achieving comparable or better results than current state-of-the-art methods for each data category within each dataset. These results confirm that TabNAT is not only suitable for unconditional generation but also applicable to other conditional generation tasks, demonstrating a wide range of application values.

7. Visualizations

In this section, we visualize the synthetic data to demonstrate that the proposed TabNAT can generate synthetic data that closely resembles the ground-truth distribution. In Figure 7 and Figure 8, we plot the 2D joint distribution of two columns of the Adult and California datasets to investigate if the ground-truth joint distribution density can be learned by the synthetic data. In Figure 9, we further plot the heatmaps of the estimation error of column pair correlations. These results visually demonstrate that TabNAT can generate synthetic samples very close to the distribution of real data and faithfully reflect the correlations between different columns of the data.

8. Conclusion

We present TabNAT, a generative model that nests a diffusion model within a bidirectional transformer framework used for heterogeneous tabular data synthesis. TabNAT uses a conditional diffusion loss and traditional cross-entropy loss to learn the conditional distributions of continuous columns and discrete columns, respectively, enabling a single model to generate both continuous and discrete features simultaneously. Furthermore, TabNAT employs masked bidirectional attention to simulate arbitrary autoregressive orders, allowing the model to generate in any direction. This not only enhances the accuracy of joint probability modeling but also enables more flexible conditional generation.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgement

This work is supported in part by NSF under grants III-2106758, and POSE-2346158.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Alaa, A., Van Breugel, B., Saveliev, E. S., and van der Schaar, M. How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. In *International Conference on Machine Learning*, pp. 290–306. PMLR, 2022.

Assefa, S. A., Dervovic, D., Mahfouz, M., Tillman, R. E., Reddy, P., and Veloso, M. Generating synthetic data in finance: Opportunities, challenges and pitfalls. In *Proceedings of the First ACM International Conference on AI in Finance*, ICAIF '20. Association for Computing Machinery, 2021. ISBN 9781450375849.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and Van Den Berg, R. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Borisov, V., Sessler, K., Leemann, T., Pawelczyk, M., and Kasneci, G. Language models are realistic tabular data generators. In *International Conference on Learning Representations*, 2023.

Castellon, R., Gopal, A., Bloniarz, B., and Rosenberg, D. Dp-tbart: A transformer-based autoregressive model for differentially private tabular data generation. *arXiv preprint arXiv:2307.10430*, 2023.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Dosovitskiy, A. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Du, T., Melis, L., and Wang, T. Remasker: Imputing tabular data with masked autoencoding. In *International Conference on Learning Representations*, 2024.

Fonseca, J. and Bacao, F. Tabular and latent space synthetic data generation: a literature review. *Journal of Big Data*, 10(1):115, 2023.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Gulati, M. S. and Roysdon, P. F. Tabmt: generating tabular data with masked transformers. In *Advances in Neural Information Processing Systems*, pp. 46245–46254, 2023.

Hernandez, M., Epelde, G., Alberdi, A., Cilla, R., and Rankin, D. Synthetic data generation for tabular health records: A systematic review. *Neurocomputing*, 493: 28–45, 2022.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, pp. 6840–6851, 2020.

Jarrett, D., Cebere, B. C., Liu, T., Curth, A., and van der Schaar, M. Hyperimpute: Generalized iterative imputation with automatic model selection. In *International Conference on Machine Learning*, pp. 9916–9937. PMLR, 2022.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, pp. 26565–26577, 2022.

Kim, J., Lee, C., and Park, N. Stasy: Score-based tabular data synthesis. In *International Conference on Learning Representations*, 2023.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

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

Kolesnikov, A., Susano Pinto, A., Beyer, L., Zhai, X., Harmsen, J., and Houlsby, N. Uvim: A unified modeling approach for vision with learned guiding codes. *Advances in Neural Information Processing Systems*, 35: 26295–26308, 2022.

Kotelnikov, A., Baranchuk, D., Rubachev, I., and Babenko, A. Tabddpm: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*, pp. 17564–17579. PMLR, 2023.

Lee, C., Kim, J., and Park, N. Codi: Co-evolving contrastive diffusion models for mixed-type tabular synthesis. In *International Conference on Machine Learning*, pp. 18940–18956. PMLR, 2023.

Li, T., Tian, Y., Li, H., Deng, M., and He, K. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37:56424–56445, 2024a.

Li, Y., Bornschein, J., and Chen, T. Denoising autoregressive representation learning. In *International Conference on Machine Learning*. PMLR, 2024b.

Liu, T., Qian, Z., Berrevoets, J., and van der Schaar, M. Goggle: Generative modelling for tabular data by learning relational structure. In *International Conference on Learning Representations*, 2023.

Lou, A., Meng, C., and Ermon, S. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, pp. 32819–32848. PMLR, 2024.

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Van Gool, L. Repaint: Inpainting using denoising diffusion probabilistic models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11461–11471, 2022.

Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Muzellec, B., Josse, J., Boyer, C., and Cuturi, M. Missing data imputation using optimal transport. In *International Conference on Machine Learning*, pp. 7130–7140. PMLR, 2020.

Pujianto, U., Wibawa, A. P., Akbar, M. I., et al. K-nearest neighbor (k-nn) based missing data imputation. In *2019 5th International Conference on Science in Information Technology (ICSITech)*, pp. 83–88. IEEE, 2019.

Razavi, A., van den Oord, A., and Vinyals, O. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pp. 14866–14876, 2019.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Tian, K., Jiang, Y., Yuan, Z., Peng, B., and Wang, L. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024.

Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

van den Oord, A., Vinyals, O., and Kavukcuoglu, K. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pp. 6309–6318, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K. Modeling tabular data using conditional gan. In *Advances in Neural Information Processing Systems*, pp. 7335–7345, 2019.

You, J., Ma, X., Ding, Y., Kochenderfer, M. J., and Leskovec, J. Handling missing data with graph representation learning. *Advances in Neural Information Processing Systems*, 33:19075–19087, 2020.

Zhang, H., Fang, L., and Yu, P. S. Unleashing the potential of diffusion models for incomplete data imputation. *arXiv preprint arXiv:2405.20690*, 2024a.

Zhang, H., Zhang, J., Shen, Z., Srinivasan, B., Qin, X., Faloutsos, C., Rangwala, H., and Karypis, G. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *International Conference on Learning Representations*, 2024b.

Zheng, S. and Charoenphakdee, N. Diffusion models for missing value imputation in tabular data. *arXiv preprint arXiv:2210.17128*, 2022.

A. Diffusion SDEs

This paper adopts the simplified version of the Variance-Exploding SDE in (Song et al., 2021). Song et al. (2021) has proposed the following general-form forward SDE:

$$\mathrm{d}\mathbf{x} = \boldsymbol{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\ \mathrm{d}\boldsymbol{w}_t = \mathrm{d}\mathbf{x} = f(t)\ \mathbf{x}\ \mathrm{d}t + g(t)\ \mathrm{d}\boldsymbol{w}_t. \tag{9}$$

Then the conditional distribution of $\mathbf{x}_t$ given $\mathbf{x}_0$ (named as the perturbation kernel of the SDE) could be formulated as:

$$p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; s(t)\mathbf{x}_0, s^2(t)\sigma^2(t)\boldsymbol{I}), \tag{10}$$

where

$$s(t) = \exp\left(\int_0^t f(\xi)\mathrm{d}\xi\right), \text{ and } \sigma(t) = \sqrt{\int_0^t \frac{g^2(\xi)}{s^2(\xi)}\mathrm{d}\xi}. \tag{11}$$

Therefore, the forward diffusion process could be equivalently formulated by defining the perturbation kernels (via defining appropriate $s(t)$ and $\sigma(t)$).

Variance Exploding (VE) implements the perturbation kernel Eq. 10 by setting $s(t) = 1$, indicating that the noise is directly added to the data rather than weighted mixing. Therefore, the noise variance (the noise level) is totally decided by $\sigma(t)$. When $s(t) = 1$, the perturbation kernels become:

$$p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \mathbf{0}, \sigma^2(t)\mathbf{I}) \ \Rightarrow\ \mathbf{x}_t = \mathbf{x}_0 + \sigma(t)\boldsymbol{\varepsilon}, \tag{12}$$

which aligns with the forward diffusion process in Eq. 9.

The sampling process of diffusion SDE is given by:

$$\mathrm{d}\mathbf{x} = [\boldsymbol{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w}_t. \tag{13}$$

For VE-SDE, $s(t) = 1 \Leftrightarrow \boldsymbol{f}(\mathbf{x}, t) = f(t) \cdot \mathbf{x} = \mathbf{0}$, and

$$\begin{aligned} &\sigma(t) = \sqrt{\int_0^t g^2(\xi)\mathrm{d}\xi} \Rightarrow \int_0^t g^2(\xi)\mathrm{d}\xi = \sigma^2(t), \\ &g^2(t) = \frac{\mathrm{d}\sigma^2(t)}{\mathrm{d}t} = 2\sigma(t)\dot{\sigma}(t), \\ &g(t) = \sqrt{2\sigma(t)\dot{\sigma}(t)}. \end{aligned} \tag{14}$$

Plugging $g(t)$ into Eq. 13, the reverse process is recovered as:

$$\mathrm{d}\mathbf{x}_t = -2\sigma(t)\dot{\sigma}(t)\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)\mathrm{d}t + \sqrt{2\sigma(t)\dot{\sigma}(t)}\mathrm{d}\boldsymbol{\omega}_t. \tag{15}$$

The conditional version of the reverse process shown in Eq. 7 is obtained via replacing $\nabla_{\mathbf{x}_t} p(\mathbf{x}_t)\mathrm{d}t$ with $\nabla_{\mathbf{x}_t} p(\mathbf{x}_t|\mathbf{z})\mathrm{d}t$.

B. Detailed Experimental Setups

B.1. Hardware Specification and Environment

We run our experiments on a single machine with Intel i9-14900K, Nvidia RTX 4090 GPU with 24 GB memory. The code is written in Python 3.10.14 and we use PyTorch 2.2.2 on CUDA 12.2 to train the model on the GPU.

B.2. Datasets

The dataset used in this paper could be automatically downloaded using the script in the provided code. We use 10 tabular datasets from Kaggle[2] or UCI Machine Learning Repository[3]: Adult[4], Default[5], Shoppers[6], Magic[7], Beijing[8], and News[9], California[10], Letter[11], Car[12], and Nursery[13], which contains varies number of numerical and categorical features. The statistics of the datasets are presented in Table 6.

In Table 6, # Rows denote the number of rows (records) in the table. # Continuous and # Discrete denote the number of continuous features and discrete features, respectively. Note that there is an additional # Target column. The target columns are either continuous or discrete, depending on the task type. All datasets (except Adult) are split into training and testing sets with the ratio $9:1$ with a fixed random seed. As Adult has its official testing set, we directly use it as the testing set. For Machine Learning Efficiency (MLE) evaluation, the training set will be further split into training and validation split with the ratio $8:1$.

B.3. Detailed Model Architectures

In this section, we introduce the detailed architecture of TabNAT.

Data Preprocessing. Then continuous columns are transformed to follow a normal distribution by QuantileTrans-

[2] https://www.kaggle.com
[3] https://archive.ics.uci.edu/datasets
[4] https://archive.ics.uci.edu/dataset/2/adult
[5] https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients
[6] https://archive.ics.uci.edu/dataset/468/online+shoppers+purchasing+intention+dataset
[7] https://archive.ics.uci.edu/dataset/159/magic+gamma+telescope
[8] https://archive.ics.uci.edu/dataset/381/beijing+pm2+5+data
[9] https://archive.ics.uci.edu/dataset/332/online+news+popularity
[10] https://www.kaggle.com/datasets/camnugent/california-housing-prices
[11] https://archive.ics.uci.edu/dataset/59/letter+recognition
[12] https://archive.ics.uci.edu/dataset/19/car+evaluation
[13] https://archive.ics.uci.edu/dataset/76/nursery

Table 6: Dataset statistics.

Dataset	# Rows	# Continuous	# Discrete	# Target	# Train	# Test	MLE Task
California	20,640	9	-	1	18,390	2,520	Classification
Letter	20,000	16	-	1	18,000	2,000	Classification
Car	1,728	-	7	1	1,555	173	Classification
Nursery	12,960	-	9	1	11,664	1,296	Classification
Adult	32,561	6	8	1	22,792	16,281	Classification
Default	30,000	14	10	1	27,000	3,000	Classification
Shoppers	12,330	10	7	1	11,098	1,232	Classification
Magic	19,021	10	1	1	17,118	1,903	Classification
Beijing	43,824	7	5	1	39,441	4,383	Regression
News	39,644	46	2	1	35,679	3,965	Regression

former[14] and discrete columns are encoded as integers by OrdinalEncoder[15]. Finally, we normalize the continuous features to have zero mean and unit variance.

B.3.1. TRANSFORMER PART

Embedding layer We use an embedding layer to transform the continuous features and each discrete feature into the same embedding dimension d.

- For the continuous features $\mathbf{x}_{\texttt{cts}} \in \mathbb{R}^{D_{\texttt{cts}}}$, we use a simple two-layer MLP to map $\mathbf{x}_{\texttt{cts}} \in \mathbb{R}^{D_{\texttt{cts}}}$ to $\mathbf{z}_{\texttt{cts}} \in \mathbb{R}^d$. The MLP has hidden dimensions $D_{\texttt{cts}} \to D_{\texttt{cts}} \to d \to d$ with `SiLU` activations.

- For each discrete column $\mathbf{x}^i_{\texttt{dst}}$, we assign a learnable embedding vector $\in \mathbb{R}^d$ for every possible category.

Transformer layers After embedding and masking, we add column-wise positional encoding to each token embedding. Furthermore, we append the `[pad]` token embedding at the beginning of the obtained data sequence. The proposed data will be further processed by a series of Transformer blocks.

We use ViT (Dosovitskiy, 2021) as the backbone of the Transformer layers, which consists of multiple Transformer blocks (Vaswani et al., 2017). Each Transformer block contains a multi-head self-attention mechanism and a feed-forward network. Specifically, we use a stack of six Transformer blocks with four attention heads.

[14] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html

[15] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html

Predictors. Given the output embeddings from the Transformer layers, i.e., $\mathbf{z}_{\texttt{cts}}$ and $\mathbf{z}^i_{\texttt{dst}}$, we predict the distribution of the corresponding column(s) conditioned on $\mathbf{z}$

For each discrete column, we apply a simple MLP predictor $\texttt{MLP}^i_{\texttt{dst}}(\cdot)$, each being a two-layer MLP with hidden dimensions $d \to d \to |\mathcal{C}^i_{\texttt{dst}}|$ with SiLU activation. $|\mathcal{C}^i_{\texttt{dst}}|$ is the number of possible categories of the i-th discrete columns.

For the continuous columns, $\mathbf{z}_{\texttt{cts}}$ will be used as the condition of the diffusion model, and we defer this part to Appendix B.3.2.

B.3.2. DIFFUSION PART

In this section, we introduce the architecture of the diffusion model. In a nutshell, we use simple MLPs as our denoising neural network, which is similar to the design in Zhang et al. (2024b) and (Kotelnikov et al., 2023), the only difference is our denoising network takes an additional input $\mathbf{z}^i$ as the conditional information.

Encoders. Given $\mathbf{x}^t_{\texttt{cts}} \in \mathbb{R}^{D_{\texttt{cts}}}$, $\mathbf{z}_{\texttt{cts}} \in \mathbb{R}^d$, and $t \in \mathbb{R}$, we use several encoders to map them into the same dimension $d_{\texttt{diff}}$ for diffusion.

- For $\mathbf{x}^t_{\texttt{cts}} \in \mathbb{R}^{D_{\texttt{cts}}}$, x-encoder is a linear transformation from $\mathbb{R}^{D_{\texttt{cts}}}$ to $\mathbb{R}^{d_{\texttt{diff}}}$, $\mathbf{x}^t_{\texttt{dn}} = x - \text{encoder}(\mathbf{x}^t_{\texttt{cts}})$
- For $\mathbf{z}_{\texttt{cts}} \in \mathbb{R}^d$, z-encoder is a linear transformation from $\mathbb{R}^d$ to $\mathbb{R}^{d_{\texttt{diff}}}$, $\mathbf{z}_{\texttt{dn}} = z - \text{encoder}(\mathbf{z}_{\texttt{cts}})$
- For $t \in \mathbb{R}$, we use sinusoidal timestep embedding of dimension $d_{\texttt{diff}}$ directly. $\mathbf{t}_{\texttt{dn}} = \text{sinusoidal}(t)$

Denoising MLP. The denoising neural network takes the summation of $\mathbf{x}^t_{\texttt{dn}}, \mathbf{z}_{\texttt{dn}}$, and $\mathbf{t}_{\texttt{dn}}$ as input. The neural network is a four-layer MLP with `SiLU` activations, and has the following hidden dimensions $d_{\texttt{diff}} \to 2 \times d_{\texttt{diff}} \to 2 \times d_{\texttt{diff}} \to d_{\texttt{diff}}$.

B.4. Hyperparameters.

TabNAT uses a fixed set of hyperparameters for all datasets. Table 7 shows the hyperparameters. Our experiments show that TabNAT is robust to the choice of hyperparameters, saving the time of meticulous hyperparameter tuning for each dataset.

Type	Parameter	Value
Training	optimizer	Adam
	initial learning rate	1e-3
	weight decay	1e-6
	LR scheduler	ReduceLROnPlateau
	training epochs	5000
	batch size	1024
Transformers	#Transformer blocks	6
	embedding dim d	32
	diffusion dimension $d_{\texttt{diff}}$	512
	#heads	4

Table 7: Default hyperparameter setting of TabNAT .

B.5. Missing Data Imputation

We formulate missing data imputation as a conditional generation task. For each data example with missing entries, we use TabNAT to sample all the missing entries for 20 times, then use the average of the 20 samples as the predicted imputations.

For a data sample of missing entries, we generate its conditional example following a random order. At each step, if the target position is a discrete distribution, we directly sample from the categorical distribution given by $\mathbf{z}^i_{\texttt{dst}}$:

$$\texttt{Softmax}(\texttt{MLP}^i_{\texttt{dst}}(\mathbf{z}^i_{\texttt{dst}})) \tag{16}$$

If a continuous column is missing, we mask the entire continuous columns to generate $\mathbf{z}_{\texttt{cts}}$, and sample the missing continuous columns using RePaint technique (Lugmayr et al., 2022; Zhang et al., 2024b).

Let's denote $\mathbf{x}_{\texttt{cts}}$ known continuous columns as $\mathbf{x}_{\text{known}}$, and the missing ones as $\mathbf{x}_{\text{unknown}}$, the reverse step is modified as a mixture of the known part's forwarding and the unknown part's denoising:

$$\begin{aligned}
\mathbf{x}^{t_{i-1}}_{\text{known}} &= \mathbf{x}_{\texttt{cts}} + \sigma(t_{i-1})\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \\
\mathbf{x}^{t_{i-1}}_{\text{unknown}} &= \mathbf{x}^{t_i}_{\texttt{cts}} + \int_{t_i}^{t_{i-1}} \mathrm{d}\mathbf{x}^{t_i}_{\texttt{cts}}, \\
\mathbf{x}^{t_{i-1}}_{\texttt{cts}} &= (1-\boldsymbol{m}) \odot \mathbf{x}^{t_{i-1}}_{\text{known}} + \boldsymbol{m} \odot \mathbf{x}^{t_{i-1}}_{\text{unknown}}, \\
\text{where } \mathrm{d}\mathbf{x}^t_{\texttt{cts}} &= -\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}^t_{\texttt{cts}}} \log p(\mathbf{x}^t_{\texttt{cts}}|\mathbf{z}_{\texttt{cts}})\mathrm{d}t \\
&\quad + \sqrt{\dot{\sigma}(t)\sigma(t)}\mathrm{d}\boldsymbol{\omega}_t,
\end{aligned} \tag{17}$$

and $\boldsymbol{m}$ is the mask indicator vector, whose 1-entries indicate the unknown locations while 0-entries indicate the known locations.

The values of unknown continuous columns can be gradually sampled via the reverse process in Eq. 17, conditioned on the known continuous columns.

B.6. Baseline Implementations

Tab-MT and DP-TBART. Tab-MT (Gulati & Roysdon, 2023) and DP-TBART (Castellon et al., 2023) are two recently proposed tabular data generation models based on MAE. To handle numerical features (with continuous distribution), Tab-MT quantizes the numerical features into 100 uniform bins, and DP-TBART quantizes the numerical features into 100 bins where each bin has the same nearest center determined by K-means. Additionally, DP-TBART employs DP-SGD (Abadi et al., 2016) to enhance the differential privacy performance. Since the focus of this paper is not on differential privacy, in our implementation, we use Adam (Kingma & Ba, 2015) optimizer.

Other Baselines. The implementations of SMOTE (Chawla et al., 2002), CTGAN (Xu et al., 2019), TVAE (Xu et al., 2019), GOOGLE[16] (Liu et al., 2023), GReaT (Borisov et al., 2023), CoDi (Lee et al., 2023), STaSy (Kim et al., 2023), TabDDPM (Kotelnikov et al., 2023), TabSyn (Zhang et al., 2024b) follows the codebase of Zhang et al. (2024b)[17].

B.7. Metrics

Most of the metrics (including Marginal, Joint, α-Precision, β-Recall, C2ST, MLE, and DCR) used in this paper directly follow the setups in Zhang et al. (2024b). Here is a reference:

- Marginal: Appendix E.3.1 in Zhang et al. (2024b).
- Joint: Appendix E.3.2 in Zhang et al. (2024b).
- α-Precision and β-Recall: Appendix F.2 in Zhang et al. (2024b).
- C2ST: Appendix F.3 in Zhang et al. (2024b).
- MLE: Appendix E.4 in Zhang et al. (2024b).
- DCR: Appendix F.6 in Zhang et al. (2024b).

Below is a summary of how these metrics work.

[16] We find the result of GOOGLE is hard to reproduce due to memory issues, so we directly use the results in Zhang et al. (2024b)

[17] https://github.com/amazon-science/tabsyn/tree/main/baselines

B.7.1. MARGINAL DISTRIBUTION

The **Marginal** metric evaluates if each column's marginal distribution is faithfully recovered by the synthetic data. We use Kolmogorov-Sirnov Test for continuous data and Total Variation Distance for discrete data.

Kolmogorov-Sirnov Test (KST) Given two (continuous) distributions $p_r(x)$ and $p_s(x)$ (r denotes real and s denotes synthetic), KST quantifies the distance between the two distributions using the upper bound of the discrepancy between two corresponding Cumulative Distribution Functions (CDFs):

$$\text{KST} = \sup_x |F_r(x) - F_s(x)|, \tag{18}$$

where $F_r(x)$ and $F_s(x)$ are the CDFs of $p_r(x)$ and $p_s(x)$, respectively:

$$F(x) = \int_{-\infty}^{x} p(x)\mathrm{d}x. \tag{19}$$

Total Variation Distance (TVD) TVD computes the frequency of each category value and expresses it as a probability. Then, the TVD score is the average difference between the probabilities of the categories:

$$\text{TVD} = \frac{1}{2}\sum_{\omega\in\Omega} |R(\omega) - S(\omega)|, \tag{20}$$

where ω describes all possible categories in a column Ω. $R(\cdot)$ and $S(\cdot)$ denotes the real and synthetic frequencies of these categories.

B.7.2. JOINT DISTRIBUTION

The **Joint** metric evaluates if the correlation of every two columns in the real data is captured by the synthetic data.

Pearson Correlation Coefficient The Pearson correlation coefficient measures whether two continuous distributions are linearly correlated and is computed as:

$$\rho_{x,y} = \frac{\text{Cov}(x,y)}{\sigma_x\sigma_y}, \tag{21}$$

where x and y are two continuous columns. Cov is the covariance, and σ is the standard deviation.

Then, the performance of correlation estimation is measured by the average differences between the real data's correlations and the synthetic data's corrections:

$$\text{Pearson Score} = \frac{1}{2}\mathbb{E}_{x,y}|\rho^R(x,y) - \rho^S(x,y)|, \tag{22}$$

where $\rho^R(x,y)$ and $\rho^S(x,y))$ denotes the Pearson correlation coefficient between column x and column y of the real data and synthetic data, respectively. As $\rho \in [-1,1]$, the average score is divided by 2 to ensure that it falls in the range of $[0,1]$, then the smaller the score, the better the estimation.

Contingency similarity For a pair of categorical columns A and B, the contingency similarity score computes the difference between the contingency tables using the Total Variation Distance. The process is summarized by the formula below:

$$\text{Contingency Score} = \frac{1}{2}\sum_{\alpha\in A}\sum_{\beta\in B} |R_{\alpha,\beta} - S_{\alpha,\beta}|, \tag{23}$$

where α and β describe all the possible categories in column A and column B, respectively. $R_{\alpha,\beta}$ and $S_{\alpha,\beta}$ are the joint frequency of α and β in the real data and synthetic data, respectively.

B.7.3. α-PRECISION AND β-RECALL

α-Precision and β-Recall are two sample-level metrics quantifying how faithful the synthetic data is proposed in Alaa et al. (2022). In general, α-Precision evaluates the fidelity of synthetic data – whether each synthetic example comes from the real-data distribution, β-Recall evaluates the coverage of the synthetic data, e.g., whether the synthetic data can cover the entire distribution of the real data (In other words, whether a real data sample is close to the synthetic data).

B.7.4. CLASSIFIER-TWO-SAMPLE-TEST (C2ST)

C2ST studies how difficult it is to distinguish real data from synthetic data, therefore evaluating whether synthetic data can recover real data distribution. The C2ST metric used in this paper is implemented by the SDMetrics[18] package.

B.7.5. MACHINE LEARNING EFFICIENCY (MLE)

In MLE, each dataset is first split into the real training and testing set. The generative models are learned on the real training set. After the models are learned, a synthetic set of equivalent size is sampled.

The performance of synthetic data on MLE tasks is evaluated based on the divergence of test scores using the real and synthetic training data. Therefore, we first train the machine learning model on the real training set, split into training and validation sets with a $8:1$ ratio. The classifier/regressor is trained on the training set, and the optimal hyperparameter setting is selected according to the performance on the validation set. After the optimal hyperparameter setting is obtained, the corresponding classifier/regressor is retrained on the training set and evaluated on the real testing set. We

[18] https://docs.sdv.dev/sdmetrics/metrics/metrics-in-beta/detection-single-table

create 20 random splits for training and validation sets, and the performance reported is the mean of the AUC/RMSE score over the 20 random trails. The performance of synthetic data is obtained in the same way.

B.7.6. Distance to Closest Record

We follow the 'synthetic vs. holdout' setting [19]. We initially divide the dataset into two equal parts: the first part served as the training set for training our generative model, while the second part was designated as the holdout set, which is not used for training. After completing model training, we sample a synthetic set of the same size as the training set (and the holdout set).

We then calculate the DCR scores for each sample in the synthetic set concerning both the training set and the holdout set. We further calculate the probability that a synthetic sample is closer to the training set (rather than the holdout set). When this probability is close to 50% (i.e., 0.5), it indicates that the distribution of distances between synthetic and training instances is very similar (or at least not systematically smaller) than the distribution of distances between synthetic and holdout instances, which is a positive indicator in terms of privacy risk.

C. Additional Experimental Results

In this section, we provide a more detailed empirical comparison between the proposed TabNAT and other baseline methods.

C.1. Detailed Results on the Fidelity Metrics

Note that in Table 1, we only present the average performance of each method on the six fidelity metrics across the ten datasets. In this section, we present a detailed performance comparison of each individual dataset:

- Marginal Distribution: Table 8
- Joint Correlation: Table 9
- α-Precision: Table 10
- β-Recall: Table 11
- Classifier-Two-Sample-Test: Table 12
- Jensen-Shannon Divergence: Table 13

[19] https://www.clearbox.ai/blog/2022-06-07-synthetic-data-for-privacy-\preservation-part-2

C.2. Detailed Results on Machine Learning Efficiency

Similarly, in Table 14, we present the detailed comparison table on Machine Learning Efficiency task.

Table 8: Performance comparison on the **Marginal** Distribution Density metric. Numbers represent the error rate in %, the lower the better.

Method	Continuous only		Discrete only		Heterogeous					
	California	**Letter**	**Car**	**Nursery**	**Adult**	**Beijing**	**Default**	**Magic**	**News**	**Shoppers**
interpolation										
SMOTE	0.99	0.97	1.00	0.57	1.59	1.78	1.49	1.07	5.28	2.48
VAE-based										
TVAE	5.37	16.70	24.12	9.81	24.32	25.13	9.94	4.39	18.48	23.93
GAN-based										
CTGAN	12.84	18.79	16.46	12.33	19.32	21.98	18.25	5.69	13.90	25.71
LLM-based										
GReaT	14.93	4.88	2.22	5.08	12.12	8.25	19.94	16.16	–	14.51
Diffusion-based										
STaSy	10.82	11.93	24.38	10.93	10.41	6.38	11.34	13.02	8.54	16.14
CoDi	18.98	22.62	1.53	0.65	24.84	12.54	16.54	11.64	28.13	36.48
TabDDPM	57.34	61.43	1.53	0.65	1.32	1.20	7.59	1.09	–	2.86
TabSyn	1.00	2.53	2.48	1.04	0.58	1.12	0.85	0.88	1.64	1.45
Autoregressive										
DP-TBART	3.30	4.46	1.98	0.53	1.17	2.68	5.03	3.90	6.28	3.05
Tab-MT	5.87	3.29	0.96	0.70	17.20	25.10	25.17	21.88	46.54	2.20
TabNAT	0.99	1.79	1.31	0.73	0.59	0.80	0.65	0.80	1.47	1.32

C.3. Visualizations

We present the 2D visualizations of the synthetic data generated by all baseline methods in Figure 10 and Figure 11, respectively. We also present the heat maps of all methods on the four datasets in Figure 12, Figure 13, Figure 14, and Figure 15, respectively.

Table 9: Performance comparison on the **Joint** Column Correlation metric. Numbers represent the error rate in %, the low the better.

Method	Continuous only		Discrete only		Heterogeous					
	California	**Letter**	**Car**	**Nursery**	**Adult**	**Beijing**	**Default**	**Magic**	**News**	**Shoppers**
interpolation										
SMOTE	2.70	1.19	3.16	1.21	3.56	1.53	6.93	2.84	2.87	3.53
VAE-based										
TVAE	5.85	5.28	38.66	18.34	36.65	31.12	19.37	4.46	6.45	20.12
GAN-based										
CTGAN	14.49	11.40	25.63	18.14	27.35	27.08	30.52	5.04	5.22	24.24
LLM-based										
GReaT	9.66	3.46	4.72	8.38	17.59	59.60	70.02	59.96	–	45.16
Diffusion-based										
STaSy	3.59	5.34	36.40	15.02	13.50	8.71	10.65	5.58	3.06	15.29
CoDi	6.89	5.25	3.52	1.31	22.72	6.42	67.88	6.93	10.81	20.18
TabDDPM	19.83	22.35	3.52	1.31	2.50	3.31	11.55	0.67	–	6.23
TabSyn	0.78	1.78	4.28	1.85	1.54	2.24	2.05	1.06	1.44	2.07
Autoregressive										
DP-TBART	2.52	1.94	3.54	1.19	2.50	2.55	6.70	1.73	1.60	2.83
Tab-MT	5.87	3.29	0.96	0.70	17.20	25.10	25.17	21.88	46.54	2.20
TabNAT	0.61	1.45	2.99	1.36	1.36	2.27	2.83	1.86	1.50	1.89

Table 10: Performance comparison on the α-**Precision** metric. Numbers represent $1 - \alpha$-**Precision**. The lower the better. Note that the numbers in Table 1 are in % while numbers in this table are in raw scale.

Method	Continuous only		Discrete only		Heterogeous					
	California	**Letter**	**Car**	**Nursery**	**Adult**	**Beijing**	**Default**	**Magic**	**News**	**Shoppers**
interpolation										
SMOTE	0.0173	0.0222	0.0103	0.0040	0.0729	0.0118	0.0228	0.0186	0.1256	0.0725
VAE-based										
TVAE	0.0191	0.0937	0.2322	0.0972	0.4124	0.1100	0.1610	0.0338	0.1530	0.5655
GAN-based										
CTGAN	0.2933	0.0522	0.1023	0.0677	0.2528	0.0723	0.3595	0.1106	0.0177	0.1292
LLM-based										
GReaT	0.1665	0.0891	0.0262	0.0836	0.4421	0.0168	0.1410	0.1454	1.0000	0.2112
Diffusion-based										
STaSy	0.8385	0.0158	0.3347	0.1385	0.2250	0.0350	0.1320	0.1853	0.0809	0.1656
CoDi	0.1333	0.0963	0.0323	0.0037	0.1805	0.0471	0.1722	0.1434	0.1041	0.0825
TabDDPM	0.8385	1.0000	0.0323	0.0037	0.0444	0.0130	0.0924	0.0146	1.0000	0.0807
TabSyn	0.0062	0.0998	0.0206	0.0056	0.0048	0.0153	0.0074	0.0062	0.0320	0.0084
Autoregressive										
DP-TBART	0.0256	0.0427	0.0093	0.0054	0.0054	0.0049	0.0727	0.0160	0.0149	0.0145
Tab-MT	0.0239	0.0415	0.0140	0.0146	0.1776	0.4839	0.0581	0.5507	0.9369	0.0158
TabNAT	0.0080	0.0383	0.0190	0.0052	0.0133	0.0032	0.0073	0.0050	0.0784	0.0233

Table 11: Performance comparison on the β-**Recall** metric. Numbers represent $1 - \beta$-**Recall**. The lower the better. Note that the numbers in Table 1 are in $\%$ while numbers in this table are in raw scale.

Method	Continuous only		Discrete only		Heterogeous					
	California	**Letter**	**Car**	**Nursery**	**Adult**	**Beijing**	**Default**	**Magic**	**News**	**Shoppers**
Interpolation										
SMOTE	0.2157	0.1338	0.0045	0.0012	0.2325	0.2080	0.2390	0.1934	0.2011	0.2386
VAE-based										
TVAE	0.6512	0.8324	0.4369	0.2625	0.8969	0.9392	0.7078	0.6243	0.7413	0.7672
GAN-based										
CTGAN	0.8412	0.9903	0.9879	0.0530	0.8246	0.6230	0.8889	0.8472	0.7732	0.7441
LLM-based										
GReaT	0.5515	0.6643	0.0034	0.0024	0.5088	0.5666	0.5796	0.6509	1.0000	0.5510
Diffusion-based										
STaSy	0.9288	0.7332	0.1075	0.0029	0.6812	0.5061	0.6421	0.5686	0.6033	0.7174
CoDi	0.5998	0.4551	0.0040	0.0009	0.9032	0.4472	0.7811	0.5139	0.6505	0.8187
TabDDPM	0.9288	1.0000	0.0040	0.0009	0.5152	0.4335	0.6150	0.5206	1.0000	0.4492
TabSyn	0.5706	0.7486	0.0031	0.0007	0.5244	0.4416	0.5200	0.5197	0.5496	0.5105
Autoregressive										
DP-TBART	0.6138	0.8859	0.0038	0.0010	0.5033	0.4536	0.5562	0.6044	0.6532	0.5285
Tab-MT	0.6272	0.8385	0.0067	0.0008	0.9681	0.4930	0.9179	0.9844	1.0000	0.5105
TabNAT	0.5038	0.4562	0.0040	0.0008	0.4928	0.4249	0.5162	0.4590	0.5277	0.4561

Table 12: Performance comparison on the **C2ST** metric. Numbers represent $100 \times (1 - \textbf{C2ST})$ (i.e. in base of 10^{-2}). The lower the better.

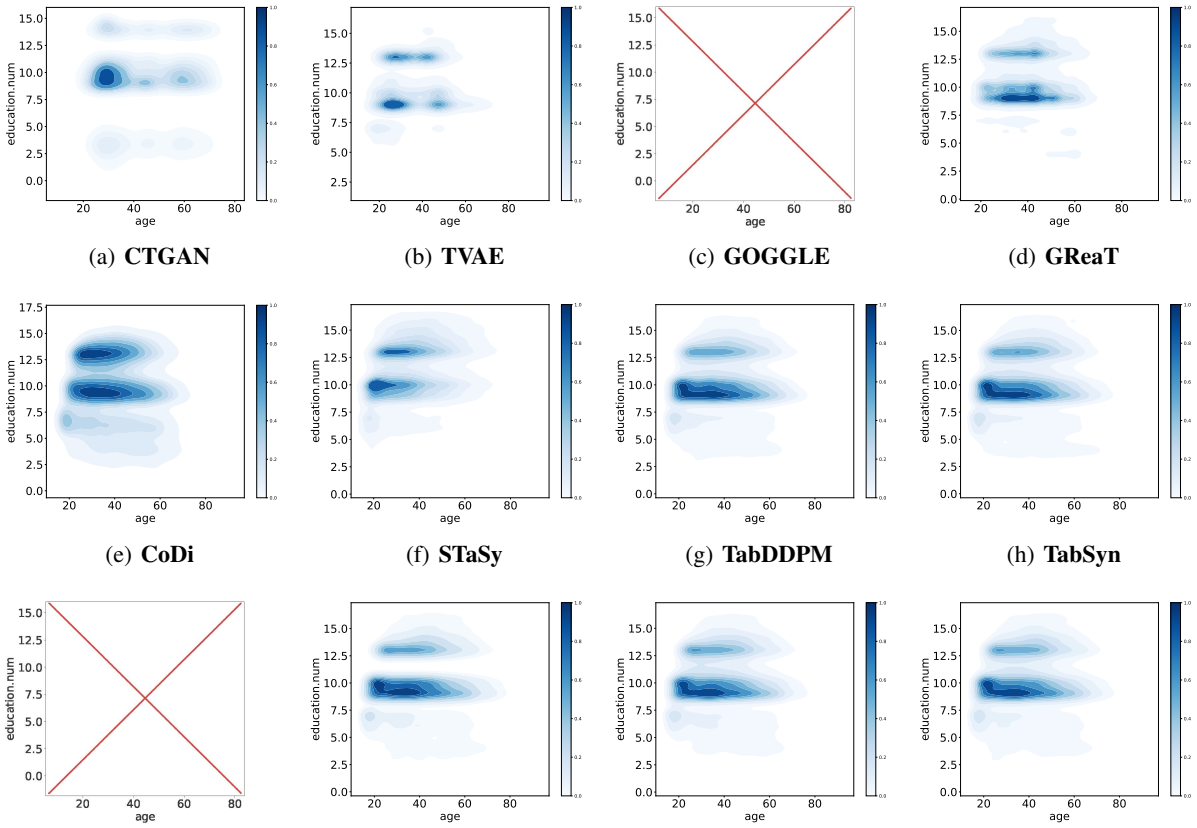
Method	Continuous only		Discrete only		Heterogeous					
	California	**Letter**	**Car**	**Nursery**	**Adult**	**Beijing**	**Default**	**Magic**	**News**	**Shoppers**
interpolation										
SMOTE	0.61	0.00	0.00	0.00	3.05	0.44	7.69	1.93	6.49	9.80
VAE-based										
TVAE	12.48	22.27	70.30	52.73	72.39	45.53	41.65	12.07	60.27	70.04
GAN-based										
CTGAN	50.11	82.40	59.59	48.63	36.79	56.82	64.60	14.15	27.64	48.86
LLM-based										
GReaT	28.38	16.14	6.05	18.41	46.24	31.07	52.90	56.74	–	57.15
Diffusion-based										
STaSy	54.61	47.75	78.52	40.43	54.02	23.48	49.29	53.97	50.21	62.20
CoDi	47.48	42.85	0.00	0.22	80.02	15.70	52.37	27.70	91.62	81.84
TabDDPM	88.01	96.86	0.00	0.22	3.95	3.29	11.75	0.95	–	16.37
TabSyn	0.71	3.18	2.90	2.88	8.05	3.60	1.33	0.08	1.77	3.05
Autoregressive										
DP-TBART	3.89	11.07	0.07	0.00	0.81	3.63	7.41	5.01	12.96	8.73
Tab-MT	8.71	9.34	0.37	0.00	99.86	99.97	91.62	74.47	100.00	1.47
TabNAT	1.27	0.55	0.00	0.74	1.30	1.55	3.43	0.1100	7.70	3.74

Table 13: Performance comparison on the Jensen-Shannon Divergence (**JSD**) metric. The lower the better. Note that the numbers in Table 1 are in % while numbers in this table are in raw scale.

Method	Continuous only		Discrete only		Heterogeous					
	California	**Letter**	**Car**	**Nursery**	**Adult**	**Beijing**	**Default**	**Magic**	**News**	**Shoppers**
interpolation										
SMOTE	0.0006	0.0006	0.0013	0.0008	0.0008	0.0003	0.0006	0.0008	0.0040	0.0015
VAE-based										
TVAE	0.0041	0.0072	0.0323	0.0132	0.0078	0.0077	0.0052	0.0036	0.0109	0.0107
GAN-based										
CTGAN	0.0182	0.0135	0.0220	0.0165	0.0038	0.0033	0.0114	0.0056	0.0078	0.0066
LLM-based										
GReaT	0.0111	0.0022	0.0030	0.0068	0.0182	0.0023	0.0076	0.0107	–	0.0056
Diffusion-based										
STaSy	0.0380	0.0057	0.0326	0.0146	0.0041	0.0030	0.0055	0.0107	0.0070	0.0086
CoDi	0.0152	0.0085	0.0020	0.0009	0.0073	0.0017	0.0067	0.0142	0.0092	0.0103
TabDDPM	0.0380	0.0382	0.0020	0.0009	0.0004	0.0092	0.0008	0.0013	–	0.0019
TabSyn	0.0006	0.0017	0.0033	0.0014	0.0004	0.0012	0.0003	0.0007	0.0016	0.0007
Autoregressive										
DP-TBART	0.0023	0.0010	0.0023	0.0008	0.0004	0.0006	0.0025	0.0018	0.0036	0.0008
Tab-MT	0.0038	0.0019	0.0013	0.0009	0.0026	0.0024	0.0034	0.0211	0.0221	0.0007
TabNAT	0.0004	0.0010	0.0018	0.0010	0.0003	0.0004	0.0007	0.0011	0.0015	0.0008

Table 14: AUC (classification task) and RMSE (regression task) scores of Machine Learning Efficiency. ↑ (↓) indicates that the higher (lower) the score, the better the performance. MLE metrics measure the relative performance difference between synthetic data and training data.

Method	Continuous only		Discrete only		Heterogeous					
	California AUC↑	**Letter** AUC↑	**Car** AUC↑	**Nursery** AUC↑	**Adult** AUC↑	**Default** AUC↑	**Shoppers** AUC↑	**Magic** AUC↑	**News** RMSE↓	**Beijing** RMSE ↓
Real data	0.999	0.989	0.999	1.000	0.927	0.770	0.926	0.946	0.842	0.423
VAE-based										
TVAE	0.986	0.989	0.746	0.939	0.846	0.744	0.898	0.912	0.979	1.010
GOGGLE	-	-	-	-	0.778	0.584	0.658	0.654	1.09	0.877
GAN-based										
CTGAN	0.925	0.729	0.899	1.000	0.874	0.736	0.868	0.874	0.895	1.065
LLM-based										
GReaT	0.996	0.983	0.979	0.999	0.913	0.755	0.902	0.888	-	0.653
Diffusion-based										
STaSy	**0.997**	0.990	0.927	0.982	0.903	0.749	0.909	0.923	0.933	0.672
CoDi	0.981	0.998	0.995	1.000	0.829	0.497	0.855	0.930	0.999	0.750
TabDDPM	0.992	0.513	0.995	1.000	0.911	0.763	0.915	0.933	-	2.665
TabSyn	0.993	0.990	0.971	0.997	0.904	0.764	0.913	0.934	0.862	0.669
Autoregressive										
DP-TBART	0.993	0.985	0.990	0.917	**0.918**	0.717	0.896	0.924	0.896	0.676
Tab-MT	0.988	0.985	0.981	**1.000**	0.873	0.714	0.912	0.822	1.002	2.098
TabNAT	0.994	**0.994**	**0.996**	0.900	0.904	**0.764**	**0.916**	**0.935**	**0.856**	**0.579**

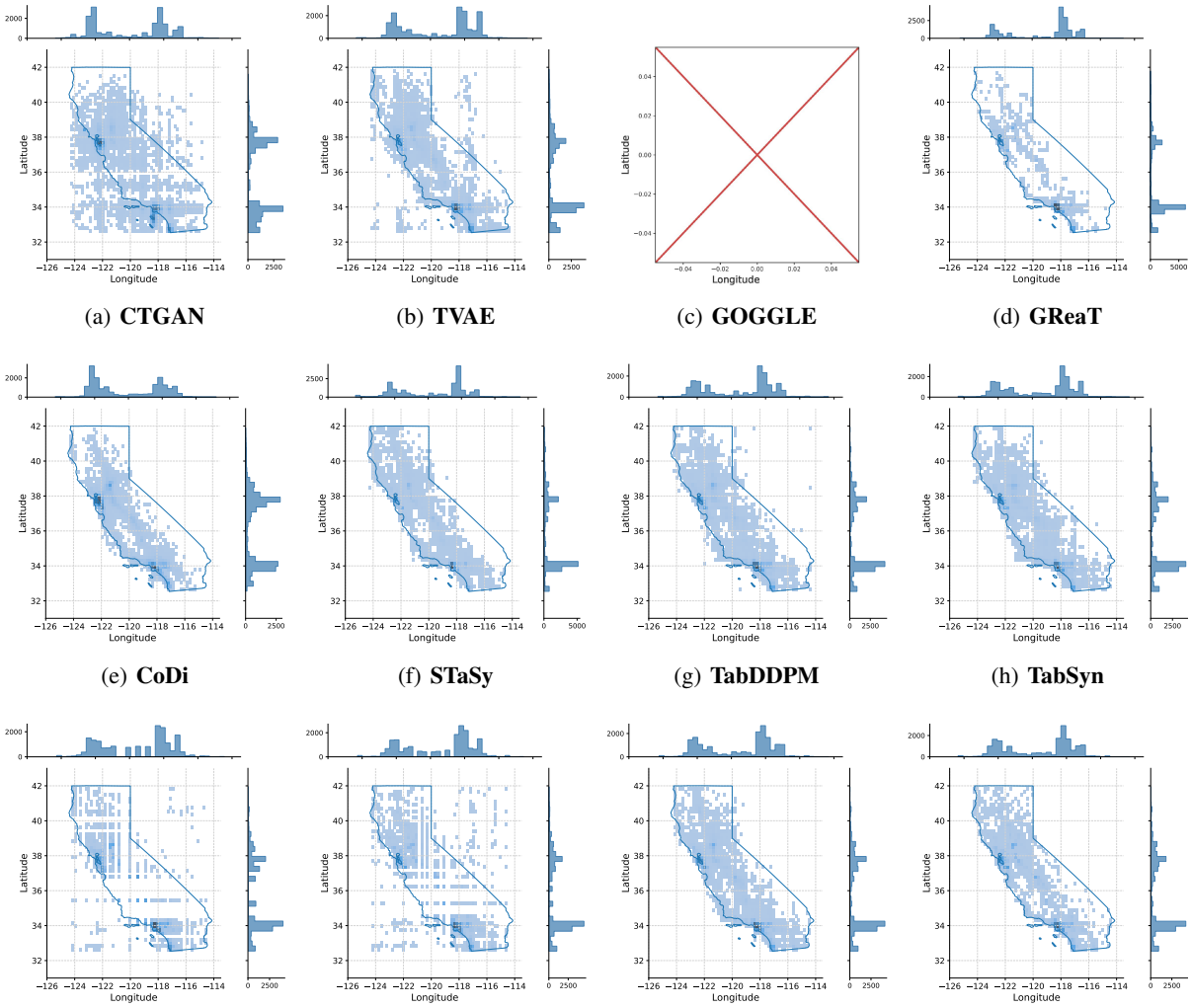


(a) **CTGAN** (b) **TVAE** (c) **GOGGLE** (d) **GReaT**

(e) **CoDi** (f) **STaSy** (g) **TabDDPM** (h) **TabSyn**

(i) **Tab-MT** (j) **DP-TBART** (k) **TabNAT (Ours)** (l) **Ground Truth**

Figure 10: Kernel density estimation (KDE) plot of the 2D joint density of 'education.num' and 'age' features in the Adult dataset. The results from GOGGLE and Tab-MT are not plotted since they either fail to generate or generate singleton synthetic data on one feature (e.g. always generate 'education.num' equals one).



(a) **CTGAN** (b) **TVAE** (c) **GOGGLE** (d) **GReaT**

(e) **CoDi** (f) **STaSy** (g) **TabDDPM** (h) **TabSyn**

(i) **Tab-MT** (j) **DP-TBART** (k) **TabNAT (Ours)** (l) **Ground Truth**

Figure 11: Scatter plots of the 2D joint density of the Longitude and Latitude features in the California Housing dataset. Blue lines represent the geographical border of California.

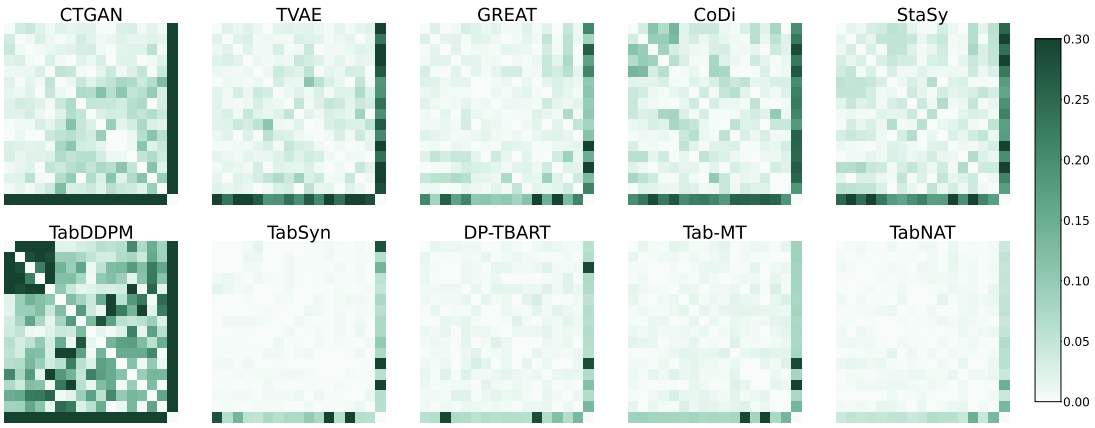


Figure 12: Heat map of synthetic data of Letter dataset.

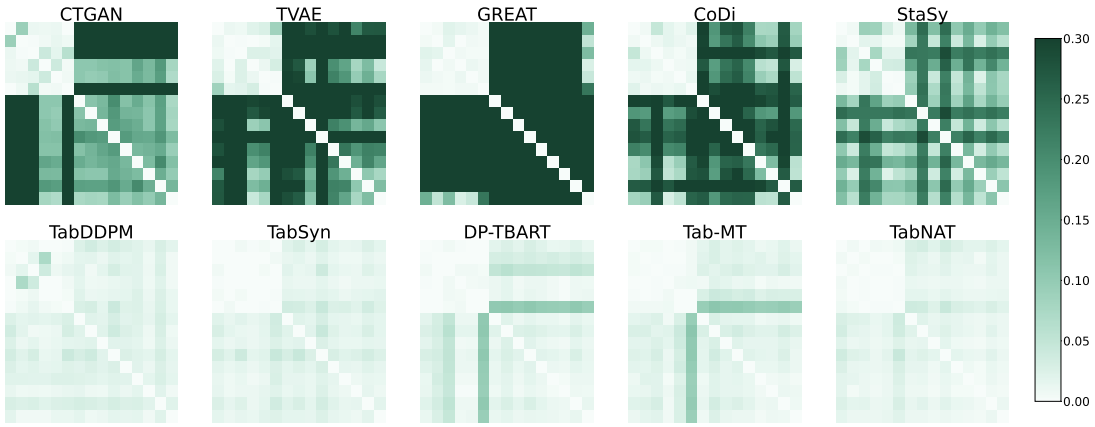


Figure 13: Heat map of synthetic data of Adult dataset.

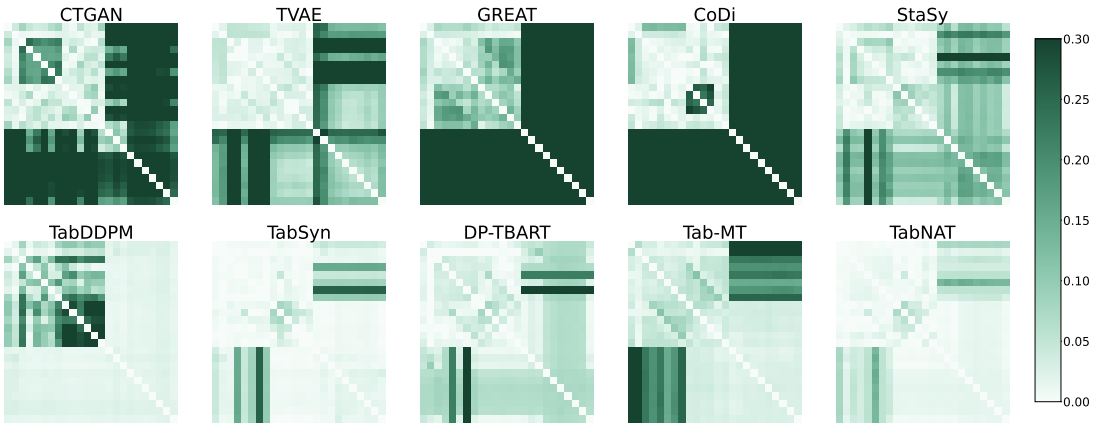


Figure 14: Heat map of synthetic data of Default dataset.

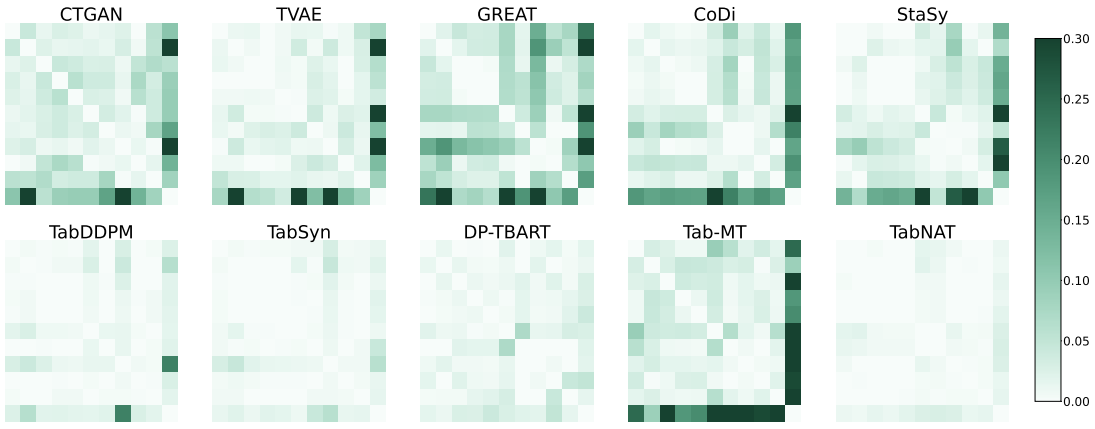


Figure 15: Heat map of synthetic data of Magic dataset.