Diffusion Models for Tabular Data Imputation and Synthetic Data Generation

MARIO VILLAIZÁN-VALLELADO, Universidad de Valladolid, Valladolid, Spain and Telefónica Scientific Research, Madrid, Spain
MATTEO SALVATORI, CARLOS SEGURA, and IOANNIS ARAPAKIS, Telefónica Scientific Research, Madrid, Spain

Data imputation and data generation have important applications across many domains where incomplete or missing data can hinder accurate analysis and decision-making. Diffusion models have emerged as powerful generative models capable of capturing complex data distributions across various data modalities such as image, audio, and time series. Recently, they have been also adapted to generate tabular data. In this article, we propose a diffusion model for tabular data that introduces three key enhancements: (1) a conditioning attention mechanism, (2) an encoder–decoder transformer as the denoising network, and (3) dynamic masking. The conditioning attention mechanism is designed to improve the model's ability to capture the relationship between the condition and synthetic data. The transformer layers help model interactions within the condition (encoder) or synthetic data (decoder), while dynamic masking enables our model to efficiently handle both missing data imputation and synthetic data generation tasks within a unified framework. We conduct a comprehensive evaluation by comparing the performance of diffusion models with transformer conditioning against state-of-the-art techniques such as Variational Autoencoders, Generative Adversarial Networks, and Diffusion Models, on benchmark datasets. Our evaluation focuses on the assessment of the generated samples with respect to three important criteria, namely: (1) machine learning efficiency, (2) statistical similarity, and (3) privacy risk mitigation. For the task of data imputation, we consider the efficiency of the generated samples across different levels of missing features. The results demonstrate average superior machine learning efficiency and statistical accuracy compared to the baselines, while maintaining privacy risks at a comparable level, particularly showing increased performance in datasets with a large number of features. By conditioning the data generation on a desired target variable, the model can mitigate systemic biases, generate augmented datasets to address data imbalance issues, and improve data quality for subsequent analysis. This has significant implications for domains such as healthcare and finance, where accurate, unbiased, and privacy-preserving data are critical for informed decision-making and fair model outcomes.

CCS Concepts: • **Computer systems organization** → *Neural networks*; • **Computing methodologies** → Temporal difference learning; **Learning latent representations**; Machine learning approaches; Machine learning; *Neural networks*;

This article has received funding from the European Union's Horizon Europe research and innovation actions under grant agreement No. 101168560 (CoEvolution). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the Commission. Neither the European Union nor the granting authority can be held responsible for them.

Authors' Contact Information: Mario Villaizán-Vallelado (corresponding author), Universidad de Valladolid, Valladolid, Spain and Telefónica Scientific Research, Madrid, Spain; e-mail: mario.villaizan@uva.es; Matteo Salvatori, Telefónica Scientific Research, Madrid, Spain; e-mail: matteo.salvatori@telefonica.com; Carlos Segura, Telefónica Scientific Research, Madrid, Spain; e-mail: carlos.seguraperales@telefonica.com; Ioannis Arapakis, Telefónica Scientific Research, Madrid, Spain; e-mail: ioannis.arapakis@telefonica.com.

ACM 1556-472X/2025/7-ART125
https://doi.org/10.1145/3742435

This is a Corrected VoR published on September 27, 2025. The VoR may still be accessed via the Supplemental Material section at https://dl.acm.org/doi/10.1145/3742435

Additional Key Words and Phrases: Data imputation, synthetic data generation, Diffusion Model, Generative Model, Transformer

Associate Editor: Xiaohui Yu

ACM Reference format:
Mario Villaizán-Vallelado, Matteo Salvatori, Carlos Segura, and Ioannis Arapakis. 2025. Diffusion Models for Tabular Data Imputation and Synthetic Data Generation. *ACM Trans. Knowl. Discov. Data.* 19, 6, Article 125 (July 2025), 32 pages.
https://doi.org/10.1145/3742435

1 Introduction

The exponential increase in data generation across sectors such as healthcare, finance, telecommunications, and energy has significantly enhanced decision-making capabilities powered by AI and **Machine Learning (ML)** technologies. However, the presence of missing or incomplete data poses significant challenges, undermining the reliability of analyses derived from ML algorithms. Moreover, the rise in strict AI regulations and data protection laws has intensified the need for robust data privacy measures, challenging traditional data handling practices.

Centralized, cloud-based ML solutions, while efficient in terms of model performance, have been repeatedly criticized for their inherent data privacy issues. Specifically, the centralized nature of these solutions necessitates the transfer of large volumes of multidimensional and privacy-sensitive user data, which raises significant privacy concerns. Moreover, centralized models contend with the issue of single point failure. Even advanced solutions like Federated Learning, which aim to decentralize data processing to enhance privacy, depend on a central server for coordinating training processes and aggregating updates. This centralization leaves systems vulnerable to potential privacy risks from information leakage attacks [17, 39], which can infer private data from shared model updates.

In addition, these ML models are particularly prone to utility loss due to missing or corrupted data, especially when dealing with sparse datasets. The handling of missing data can favor certain statistical interpretations and subsequent implications for policy and practice [9]. For instance, the mean substitution method, often used to handle missing data, may lead to inconsistent bias, especially in the presence of a great inequality in the number of missing values for different features [26]. Furthermore, when missing data are not missing at random (**Missing Not at Random (MNAR)**), even multiple imputations do not lead to valid results [16]. Such imputation methods that fill in blanks with estimated values may inadvertently lead to the creation and transmission of inaccurate or misleading information.

Current efforts to address these issues, such as the application of differential privacy or the use of specialized hardware (e.g., Trusted Execution Environments [37]), often result in a tradeoff between privacy and data utility or necessitate additional infrastructure.

Given these challenges, there is a pressing need for solutions that effectively manage data integrity and privacy. Recent advancements in ML, specifically **Generative Adversarial Networks (GANs)** [14] and Diffusion Models [18, 48], have shown promise in generating high-fidelity synthetic data that preserve the statistical properties of original datasets while mitigating privacy concerns, since they follow the original distribution without directly exposing or replicating sensitive information. These generative methods have found their way into applications like image and audio processing [35, 47, 64] and, more recently, have expanded to address tabular data as well [30, 31, 33, 62, 65].

Specifically, for tabular data, synthetic data stands out as a privacy-preserving alternative to real data that may contain personally identifiable information. It enables the generation of datasets that

mimic the statistical properties of their original counterparts, while mitigating the risk of individual privacy breaches. In addition, this approach to generating new samples can augment existing datasets by, for example, correcting class imbalances, reducing biases, or expanding their size when dealing with sparse or limited data. Furthermore, integrating methods for differential privacy [11, 23] with generative models for tabular data makes it possible to share synthetic datasets across units in large organizations, addressing legal or privacy concerns that often impede technological innovation adoption.

In this study, we consider synthetic data generation as a general case of data imputation. In instances where every column in a sample has missing values, the task of data imputation naturally transitions to synthesizing new data. We introduce *MTabGen*, a new conditioning in diffusion model for tabular data using an encoder–decoder transformer and a dynamic masking mechanism that makes it possible to tackle both tasks with a single model. During the training step of the model, the dynamic masking randomly masks features that we later generate or impute. The unmasked features are used as context or condition for the reconstruction of masked features during the reverse denoising phase of the diffusion process. We refer to the features to be denoised during the reverse denoising phase as *masked features* hereinafter.

In our analysis, we perform a rigorous evaluation of the proposed approach using several benchmark datasets, each with a wide range of features. We demonstrate that our method shows overall improved performance compared to existing baselines, particularly in handling high-dimensional datasets, thereby highlighting its robustness and adaptability in complex tabular data scenarios. The key contributions of this work are the following:

- We propose a new conditioning mechanism for tabular diffusion models. We model the interaction between condition and masked features (e.g., features to-be-denoised) by using an attention mechanism. Within this mechanism, the embedding of the masked features plays the role of query (**Q**), and the embedding of the condition plays the roles of key (**K**) and value (**V**) (see [57] for details). Compared to the standard approach, where condition and masked features are added [31, 68] or concatenated [33], our method allows to learn more complex relationships showing globally improved performances.
- We incorporate a full encoder–decoder transformer within the diffusion process as the denoising model. The encoder learns the condition embedding, while the decoder models the representation of the masked features. Using transformer layers enhances the learning of inter-feature interactions: within the condition for the encoder and the masked features for the decoder. Additionally, the encoder–decoder architecture allows the implementation of the conditioning attention mechanism explained in the previous item. To the best of our knowledge, [68] is the only prior work that has considered diffusion models for tabular data, using a transformer denoising component. However, in this article, the transformer layer is limited to learning the masked feature representation, without modeling the condition nor using conditioning attention mechanism.
- We extend the masking mechanism proposed by Zheng and Charoenphakdee [68] to train a single model capable of multitasking, handling both missing data imputation and synthetic data generation. This is facilitated by the transformer encoder–decoder architecture, which allows for arbitrary modification of the split between condition and masked features during the training phase.
- We conduct extensive experiments on several public datasets and demonstrate average performance gain over state-of-the-art baselines, for both tasks, missing data imputation and synthetic data generation. We evaluate the synthetic data with respect to three important criteria: (1) ML efficiency, (2) statistical similarity and (3) privacy risk.

2 Related Work

Tabular Data Modeling and Benchmarks. The evaluation of models for tabular data requires diverse benchmarks that reflect real-world data conditions. Several recent studies have established such benchmarks for both data imputation and supervised learning. Jäger et al. [22] present a comprehensive evaluation of data imputation methods, comparing both classical and deep learning-based imputation approaches. Their results show that imputation can significantly improve downstream ML tasks, especially when training data is fully observed. For supervised tasks, Grinsztajn et al. [15] argue that tree-based models such as XGBoost and Random Forests often outperform deep learning models due to the specific properties of tabular data, including irregular patterns and uninformative features. The authors provide a standard benchmark for evaluating both traditional and novel approaches on a variety of real-world datasets. Borisov et al. [7] also highlight the challenges faced by deep learning models when dealing with heterogeneous tabular data and emphasize the need for architectures specifically designed for tabular datasets. In this work, we extend the benchmark proposed by [7] to include a wider variety of datasets, providing a more exhaustive evaluation of generative tasks.

Diffusion Models. Originally introduced by Sohl-Dickstein et al. [48] and Ho et al. [18], diffusion models utilize a two-step generative approach. Initially, they degrade a data distribution using a forward diffusion process by continuously introducing noise from a known distribution. Subsequently, they employ a reverse process to reconstruct the original data structure. At their core, these models leverage parameterized Markov chains, starting typically from a foundational distribution such as a standard Gaussian, and use deep neural networks to reverse the diffusion. Demonstrated by recent advancements [10, 40], diffusion models have showcased their capability, potentially surpassing GANs in image generation capabilities. Recent works, such as *StaSy* [30], *CoDi* [33], *Tabsyn* [65], *TabDDPM* [31], or *TabCSDI* [68], adapt diffusion models to handle tabular data in both synthetic data generation and data imputation tasks.

Missing Data Imputation. Handling missing values in datasets is a non-trivial problem. Traditional approaches may involve excluding rows or columns with missing entries or imputing missing values using the average values of the corresponding feature. However, recent efforts have been focusing on ML techniques [4, 24, 56] and deep generative models [5, 20, 32, 60, 63], and new models using diffusion processes have been developed for data imputation tasks. Specifically, *TabCSDI*, based on the *CSDI* model originally designed for time-series data, adapts this technology for tabular data imputation. *TabCSDI* employs three common preprocessing techniques: (1) one-hot encoding, (2) analog bits encoding, and (3) feature tokenization. These methods allow it to treat continuous and categorical variables uniformly in a Gaussian diffusion process, regardless of their original types. In contrast, *MTabGen* implements a dual diffusion mechanism adapted for both continuous and categorical data, maintaining the unique statistical properties of each feature type throughout the diffusion process. As noted previously, *TabCSDI* uses a transformer layer to learn only the interactions within the masked features and then adds the transformer output to the condition embedding. In our case, *MTabGen* uses a more adaptable and general approach: the condition embedding is modeled by using a transformer encoder and then fed into a conditioning attention mechanism.

Generative Models. The application of this family of models to tabular data has been gaining increased attention within the ML community. In particular, tabular VAEs [62] and GANs [12, 29, 41, 53, 61, 66, 67] have shown promising results. Recently, *StaSy* [30], *CoDi* [33], *Tabsyn* [65], and *TabDDPM* [31] have been proposed as powerful alternatives to tabular data generation, leveraging the strengths of Diffusion Models.

Specifically, *STaSy*, applies a score-based generative approach [50], integrating self-paced learning and fine-tuning strategies to enhance data diversity and quality by stabilizing the denoising score matching training process.

CoDi addresses training challenges with mixed-data types using a dual diffusion model approach. One model handles continuous features and the other manages discrete (categorical) features. Both models use a UNet-based architecture with linear layers instead of traditional convolutional layers and are trained to condition on each other's outputs. This fixed conditioning setup is designed specifically to handle the interactions between continuous and categorical variables effectively.

Tabsyn applies the concepts of latent diffusion models [47, 55] to tabular data. First, a VAE encodes mixed-type data into a continuous latent space and then a diffusion model learns this latent distribution. The VAE uses a transformer encoder–decoder to capture feature relationships, while the diffusion model uses an MLP in the reverse denoising process. There are no transformers used in the denoising step, and it does not include a mask conditional method.

Finally, *TabDDPM* manages datasets with mixed-data types by using Gaussian diffusion for continuous features and multinomial diffusion for categorical ones. In the preprocessing step, continuous features are scaled using a min-max scaler, and categorical features are one-hot encoded. Then each type of data is sent to its specific diffusion process. After the denoising process, the preprocessing is reversed by scaling back continuous variables, and categorical ones are estimated by first applying a softmax function and then selecting the most likely category. For classification datasets, *TabDDPM* adopts a class-conditional model consisting in the addition of the condition embedding to the output of the model, while for regression datasets, it integrates target values as an additional feature. It utilizes an MLP architecture as the denoising network optimized with a hybrid objective function that includes MSE and **Kullback–Leibler (KL)** divergence to predict continuous and categorical data.

Our method is based on *TabDDPM*, following the same logic of having two separate diffusion models for continuous and categorical data. To this end, we augment the model with three key improvements. First, we employ a transformer-based encoder–decoder as the denoising model, which enhances the capability to learn inter-feature interactions for both, condition and masked features. Second, we integrate the conditioning directly into the transformer's attention mechanism rather than simply adding embeddings: this approach reduces learning bias, improving how the interaction between the condition and masked features is modeled. Third, we enable dynamic masking during training, which allows our model to handle varying numbers of visible variables, thus supporting both synthetic data generation and missing data imputation within a single framework. In the following sections, we demonstrate that these contributions lead to improved results across various datasets, generally outperforming *TabDDPM* and other state-of-the-art algorithms.

3 Background

Diffusion models, as introduced by Sohl-Dickstein et al. [48] and Ho et al. [18], involve a two-step process: first degrading a data distribution using a forward diffusion process and then restoring its structure through a reverse process. Drawing insights from non-equilibrium statistical physics, these models employ a forward Markov process which converts a complex unknown data distribution into a simple known distribution (e.g., Gaussian) and vice-versa a generative reverse Markov process that gradually transforms a simple known distribution into a complex data distribution.

More formally, the forward Markov process $q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$ gradually adds noise to an initial sample x_0 from the data distribution $q(x_0)$ sampling noise from the predefined distributions $q(x_t|x_{t-1})$ with variances $\{\beta_1, \ldots, \beta_T\}$. Here $t \in [1, T]$ is the timestep, T is the total number

of timesteps used in the forward/reverse diffusion processes and $1{:}T$ means the range of timesteps from $t = 1$ to $t = T$.

The reverse diffusion process $p(x_{0:T}) = \prod_{t=1}^{T} p(x_{t-1}|x_t)$ gradually denoises a latent variable $x_T \sim q(x_T)$ and allows generating new synthetic data. Distributions $p(x_{t-1}|x_t)$ are approximated by a neural network with parameters θ.

In this work, we use the hat notation (e.g., $\hat{p}(x_{t-1}|x_t)$) to indicate that a variable is estimated by a neural network model trained with parameters θ. Although these estimations depend on θ, we omit θ for notational simplicity. Thus, $\hat{p}(x_{t-1}|x_t)$ should be interpreted as $\hat{p}_\theta(x_{t-1}|x_t)$, where the model parameters θ influence the predicted value.

The parameters are learned optimizing a **Variational Lower Bound (VLB)**:

$$L_{\text{vlb}} := \sum_{t=0}^{T} L_t \tag{1}$$

$$L_0 := -\log \hat{p}(x_0|x_1) \tag{2}$$

$$L_{t-1} := D_{KL}(q(x_{t-1}|x_t, x_0) \,||\, \hat{p}(x_{t-1}|x_t)) \tag{3}$$

$$L_T := D_{KL}(q(x_T|x_0) \,||\, p(x_T)). \tag{4}$$

The term $q(x_{t-1}|x_t, x_0)$ is the *forward process posterior distribution* conditioned on x_t and on the initial sample x_0. L_{t-1} is the KL divergence between the posterior of the forward process and the estimated reverse diffusion process $\hat{p}(x_{t-1}|x_t)$.

Gaussian diffusion models operate in continuous spaces ($x_t \in \mathbb{R}^n$) and in this case the aim of the forward Markov process is to convert the complex unknown data distribution into a known Gaussian distribution. This is achieved by defining a forward noising process q that given a data distribution $x_0 \sim q(x_0)$ produces latents x_1 through x_T by adding Gaussian noise at time t with variance $\beta_t \in (0, 1)$:

$$\begin{aligned} q(x_t|x_{t-1}) &:= \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I\right) \\ q(x_T) &:= \mathcal{N}(x_T; 0, I). \end{aligned} \tag{5}$$

If we know the exact reverse distribution $q(x_{t-1}|x_t)$, by sampling from $x_T \sim \mathcal{N}(0, I)$, we can execute the process backward to obtain a sample from $q(x_0)$. However, given that $q(x_{t-1}|x_t)$ is influenced by the complete data distribution, we employ a neural network for its estimation:

$$\hat{p}(x_{t-1}|x_t) := \mathcal{N}\left(x_{t-1}; \hat{\mu}(x_t, t), \hat{\Sigma}(x_t, t)\right). \tag{6}$$

Ho et al. [18] proposes a simplification of Equation (6) by employing a diagonal variance $\hat{\Sigma}(x_t, t) = \sigma_t I$, where σ_t are constants dependent on time. This narrows down the prediction task to $\hat{\mu}(x_t, t)$. While a direct prediction of this term via a neural network seems the most intuitive solution, another approach could involve predicting x_0 and then leveraging earlier equations to determine $\hat{\mu}(x_t, t)$. Alternatively, it could be inferred by predicting the noise ϵ, as done by Ho et al. [18]. In this work, the authors propose the following parameterization:

$$\hat{\mu}(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\epsilon}(x_t, t)\right), \tag{7}$$

where $\hat{\epsilon}(x_t, t)$ is the prediction of the noise component ϵ used in the forward diffusion process between the timesteps $t-1$ and t, and $\alpha_t := 1 - \beta_t$, $\bar{\alpha}_t := \prod_{i \le t} \alpha_i$.

The objective Equation (1) can be finally simplified to the sum of MSEs between $\hat{\epsilon}(x_t, t)$ and ϵ over all timesteps t:

$$L_{\text{simple}} = E_{t,x_0,\epsilon}\left[||\epsilon - \hat{\epsilon}(x_t, t)||^2\right]. \tag{8}$$

For a detailed derivation of these formulas and a deeper understanding of the methodologies, readers are referred to the original paper by Ho et al. [18] and Nichol and Dhariwal [40].

Multinomial Diffusion Models. Hoogeboom et al. [19] designed the procedures for generating categorical data, where $x_t \in \{0, 1\}^{Cl}$ is a one-hot encoded categorical variable with Cl classes. Here, the aim of the forward Markov process is to convert the complex unknown data distribution into a known uniform distribution. The multinomial forward diffusion process $q(x_t|x_{t-1})$ is a categorical distribution that corrupts the data by uniform noise over Cl classes:

$$\begin{aligned} q(x_t|x_{t-1}) &:= Cat\left(x_t; (1-\beta_t)\, x_{t-1} + \beta_t/Cl\right) \\ q(x_T) &:= Cat\left(x_T; 1/Cl\right) \\ q(x_t|x_0) &:= Cat\left(x_t; \bar{\alpha}_t x_0 + (1-\bar{\alpha}_t)/Cl\right). \end{aligned} \tag{9}$$

Intuitively, at each timestep, the model updates the data by introducing a small amount of uniform noise β_t across the Cl classes combined with the previous value x_{t-1}, weighted by $(1-\beta_t)$. This process incrementally introduces noise while retaining a significant portion of the prior state, promoting the gradual transition to a uniform distribution. This noise introduction mechanism allows for the derivation of the forward process posterior distribution $q(x_{t-1}|x_t, x_0)$ from the provided equations as follows:

$$q(x_{t-1}|x_t, x_0) = Cat\left(x_{t-1}; \pi / \sum_{k=1}^{Cl} \pi_k\right), \tag{10}$$

where $\pi = [\alpha_t x_t + (1-\alpha_t)/Cl] \odot [\bar{\alpha}_{t-1} x_0 + (1-\bar{\alpha}_{t-1})/Cl]$.

The reverse distribution $\hat{p}(x_{t-1}|x_t)$ is parameterized as $q(x_{t-1}|x_t, \hat{x}_0(x_t, t))$, where $\hat{x}_0$ is predicted by a neural network. Specifically, in this approach, instead of estimating directly the noise component ϵ, we predict x_0, which is then used to compute the reverse distribution. Then, the model is trained to maximize the VLB (Equation (1)).

4 MTabGen

MTabGen shares foundational principles with the *TabDDPM* approach [31], but introduces enhancements in both the denoising model and the conditioning mechanism using a transformer encoder–decoder architecture. These improvements enhance the quality of synthetic data and strengthen the conditioning process needed for the reverse diffusion. As a result, the model is capable of generating conditioned synthetic data and performing missing data imputation, generally improving on the performance of other state-of-the-art models for these tasks.

4.1 Problem Definition

In this work, we address the challenge of modeling tabular datasets for supervised tasks. Our approach is designed to handle both numerical and categorical features, while also effectively dealing with missing values. We focus on tabular datasets for supervised tasks $D = \left\{\left(x_i^c, x_i^n, y_i\right)\right\}_{i=1}^{N}$, where $x_i^n \in \mathbb{R}^{K_{\text{num}}}$ represents the set of numerical features, $x_i^c \in \mathbb{Z}^{K_{\text{cat}}}$ represents the set of categorical features, with each categorical feature potentially having a different number of categories, and y_i is the target label for row i. The dataset contains N rows, where K_{num} and K_{cat} are the number of numerical and categorical features, respectively, with the total number of features being $K = K_{\text{num}} + K_{\text{cat}}$.

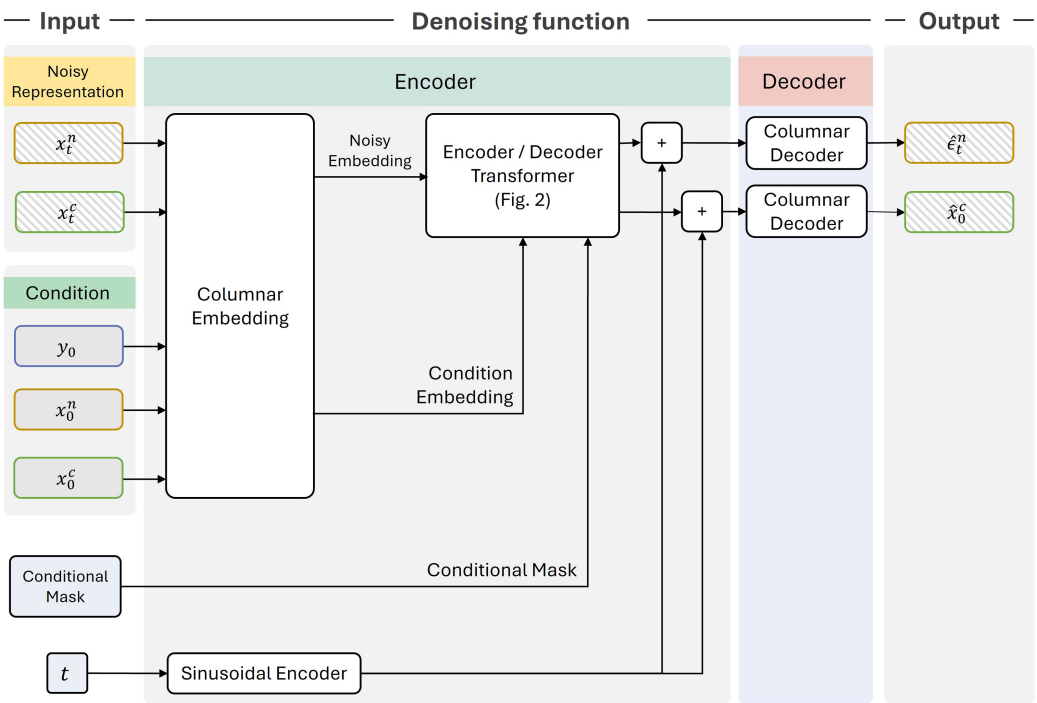

Fig. 1. Denoising function dn_θ. The function takes noisy data representations x_t^n, x_t^c, conditioning values $x^{\text{cond}} = x_0^n, x_0^c$ and conditioning target y_0, a conditional mask `Mask` and current timestep t as inputs. The `Columnar Embedding` layer projects numerical and categorical features into a shared latent space, while the `Encoder-Decoder Transformer` (detailed in Figure 2) refines this representation by modeling intra- and inter-feature relationships. Outputs are generated through a `Columnar Decoder`, predicting the estimated noise for numerical features ($\hat{\epsilon}_t^n$) and logits for categorical features ($\hat{x}_0^c$).

In our approach, we model numerical features with Gaussian diffusion and categorical features with multinomial diffusion. Each feature is subjected to a distinct forward diffusion procedure, which means that the noise components for each feature are sampled individually.

MTabGen generalizes the approach of *TabDDPM* where the model originally learned $p\left(x_{t-1}|x_t, y\right)$, i.e., the probability distribution of x_{t-1} given x_t and the target y. We extend this by allowing conditioning not only on the target y but also on a subset of input features, aligning with the strategies proposed by Zheng and Charoenphakdee [68] and Tashiro et al. [52]. Specifically, we partition variable x into two subsets: x^{mask} and x^{cond}. Here, x^{mask} contains the features masked and subjected to forward diffusion, while x^{cond} represents the untouched variable subset that conditions the reverse diffusion. This setup models $p\left(x_{t-1}^{\text{mask}}|x_t^{\text{mask}}, x^{\text{cond}}, y\right)$, with x^{cond} and y remaining constant across timesteps t. This approach not only enhances model performance in data generation, but it also enables the possibility of performing data imputation with the same model.

The reverse diffusion process $p\left(x_{t-1}^{\text{mask}}|x_t^{\text{mask}}, x^{\text{cond}}, y\right)$ is parameterized by a single neural network shown in Figures 1–3. This common neural network predicts simultaneously both the amount of noise added for numerical features between steps $t-1$ and t and the distribution of categorical features at time $t = 0$. The output dimensionality is $K_{\text{num}} + \sum_{i=1}^{K_{\text{cat}}} Cl_i$, where Cl_i denotes the number of classes for the ith categorical feature.

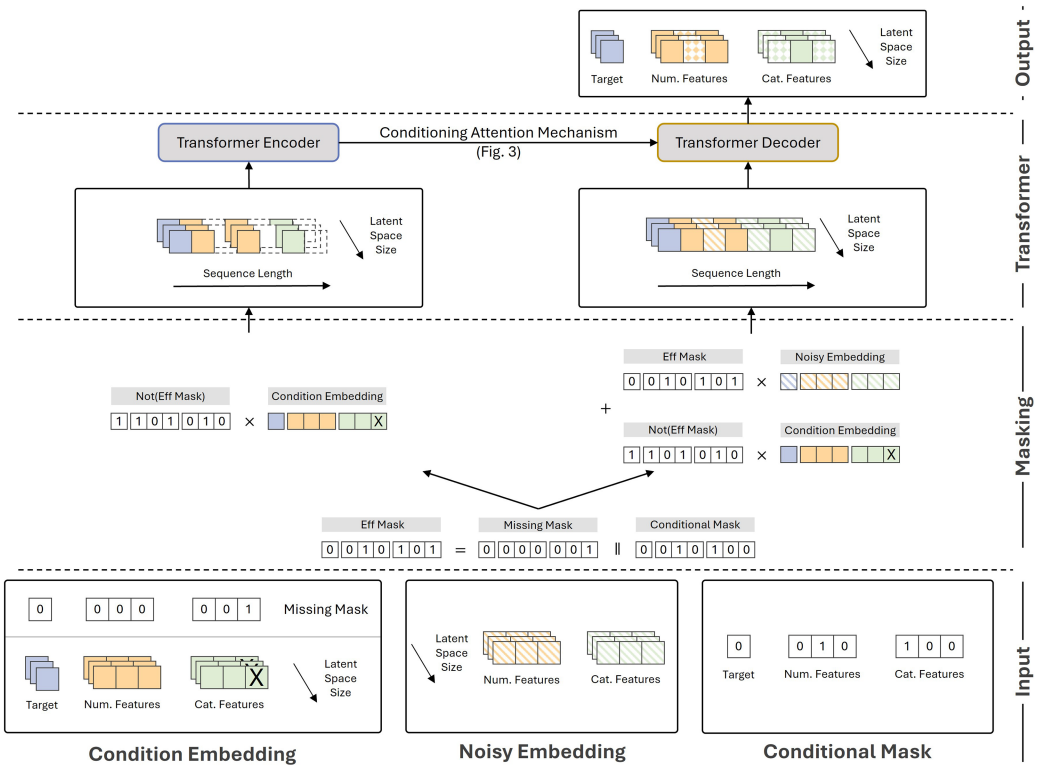

Fig. 2. Conditional transformer encoder–decoder model. The encoder takes three inputs: noisy data embedding, condition embedding, and a conditional mask (Mask). The embedding of a conditioning feature is represented by a fully filled square, while the embedding of a noisy feature is depicted as a dashed-colored square. The effective mask (EffMask) combines missing and conditional masks, allowing the encoder to process conditioning features and target that are unaffected by forward diffusion and without missing values, learning a conditioning context vector for the decoder. The decoder then refines the representation of masked features using all available information, including the encoder context, both masked and conditioning features and targets. The final representation of the masked features is depicted as squares filled with colored rhombuses.

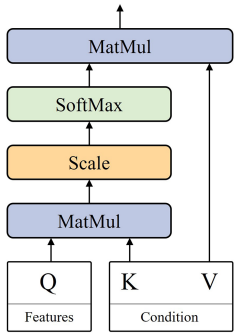

Fig. 3. Conditioning attention mechanism: The condition embedding produced by the transformer encoder is used in the decoder attention mechanism. More in detail, the condition embedding plays the roles of **K** and **V** whereas the feature embedding plays the role of **Q**.

4.2 Model Overview

MTabGen utilizes an encoder–decoder transformer architecture as the denoising function to refine the representation of noisy data and produce clean samples through reverse diffusion. The workflow is depicted in Figure 1, with the following key components: the columnar embedding, the encoder–decoder transformer, and the output decoders for each feature type.

4.2.1 Denoising Function Architecture. The denoising function dn_θ is responsible for reconstructing the original data representation from noisy samples. Figures 1–3 illustrate its detailed workflow and components. The function follows an encoder–decoder model structure and takes as inputs the current noisy representations of numerical and categorical data (x_t^n and x_t^c) at timestep t, a conditional mask (`Mask`) identifying features involved in the forward diffusion process, the conditioning values (x^{cond}), the target (y), and the current timestep (t).

The primary objective of the denoising function is to produce two outputs at each timestep: the estimated noise ($\hat{\epsilon}_t^n$) for numerical features and the logits ($\hat{x}_0^c$) for categorical features, representing their predicted original distribution. These estimated outputs are then used by Equations (7) and (10), respectively, to perform the reverse denoising at each timestep t as described in the following sections.

4.2.2 Columnar Embedding. To facilitate the learning of the denoising function within the transformer model, a *columnar embedding* projects both numerical and categorical features of x^{mask} and x^{cond} into a shared latent space. In the case of categorical features, an embedding layer maps each categorical value into a dense vector representation, whereas numerical features are embedded using a linear transformation followed by a ReLU activation function. The embedding of the target variable y depends on the supervised task. For regression, y is treated as a continuous variable, while for classification, it is processed through an embedding layer similar to other categorical features.

4.2.3 Conditional Encoder–Decoder Transformer. The workflow, illustrated in Figure 2, consists of three inputs, two intermediate stages—masking and transformer—and a final output, arranged from bottom to top. The three inputs are: (1) the columnar embedding of the noisy data, referred to as the noisy embedding, (2) the columnar embedding of the conditioning data, termed the condition embedding, and (3) the conditional mask, denoted as `Mask`. In the conditional mask, features involved in the forward diffusion process are indicated by `Mask` = 1, while features not involved are marked as `Mask` = 0. If the original dataset contains missing values, an additional mask, termed `MissingMask`, is introduced, where `MissingMask` = 1 for missing values, as shown in Figure 2.

During the masking phase, an effective mask (`EffMask`) is defined as the logical OR between the conditional mask and the missing value mask, i.e., `EffMask` = `Mask||MissingMask`. This `EffMask` is then utilized to generate the input expected by the transformer encoder and decoder.

In the transformer phase, the encoder exclusively processes the conditioning features characterized by `EffMask` = 0, which correspond to features unaffected by the forward diffusion process and those that do not contain missing values. Simultaneously, the decoder processes two distinct types of inputs: (1) a combination of noisy data, which includes features involved in the forward diffusion process or containing missing values (`EffMask` = 1), and conditioning features (`EffMask` = 0), and (2) the output from the encoder . This setup enables the encoder to provide contextual information from the conditioning features, allowing the decoder to generate refined representations of the features that require denoising. The output of the decoder corresponds to the final output stage, as shown in Figure 2.

Algorithm 1: Sampling or Reverse Diffusion Process

Inputs: Denoising function dn_θ, Conditional Mask Mask, Conditioning Value x^{cond}, y and number of Timesteps T
Preprocess the conditioning value x^{cond}, y.
Initialize noise:
    Sample initial numerical gaussian noise: $x_T^n \sim \mathcal{N}\left(x_T^n; 0, I\right)$.
    Sample initial categorical uniform noise: $x_T^c \sim Cat\left(x_T^c; 1/Cl\right)$.
for t=T, T-1, ..., 2, 1, 0 **do**
    $\hat{\epsilon}_t^n, \hat{x}_0^c = dn_\theta\left(x_t^n,\ x_t^c,\ \texttt{Mask},\ x^{\text{cond}},\ y,\ t\right)$.
    Perform a reverse Gaussian diffusion step using Eq. 7 to compute x_{t-1}^n with $\hat{\epsilon}_t^n$ and x_t^n.
    Perform a reverse multinomial diffusion step using Eq. 10 to compute x_{t-1}^c with $\hat{x}_0^c$ and x_t^c.
end for
Postprocess x_0^n and x_0^c to reverse the preprocessing from state 2.

Decoder and Attention Mechanism. The decoder attends to all conditioning features, using the context provided by the encoder. The attention mechanism, depicted in Figure 3, incorporates the output of the encoder as the key (**K**) and value (**V**) vectors, while the features, including the noisy data, serves as the query (**Q**) vector, as described in Vaswani et al. [57]. This enables the decoder to refine masked feature representations by selectively focusing on interactions between conditioning variables and noisy data.

This attention-based conditioning mechanism is more general and exhibits less learning bias compared to recent approaches in the literature on diffusion models for tabular data, such as *TabDDPM* and *TabCSDI*, where the condition embedding is only added to the masked feature embedding. This advantage holds true even in scenarios where the encoder processes only one variable (the target variable of a supervised task). Here, the transformer encoder functions similarly to an MLP, but its output continues to be used in the decoder's attention mechanism, enhancing overall performance (see discussion on ablation study in Section 6).

4.2.4 Output Denoising Parameters. Following the description of the workflow of Figure 1, the final latent representation of the features to denoise is obtained by summing the conditional encoder–decoder transformer output with the timestep embedding, which is derived by projecting the sinusoidal temporal embedding [10, 40] into the transformer embedding dimension, using a linear layer followed by the Mish activation function [36]. Last, this representation is decoded to produce the output. Each feature has its own *decoder* consisting of two successive blocks integrating a linear transformation, normalization, and ReLU activation. Depending on the feature type (numerical or categorical), an additional linear layer is appended with either a singular output for numerical features or multiple outputs, corresponding to the number of classes for categorical ones.

4.3 Sampling or Reverse Diffusion Process

The primary goal of the sampling or reverse diffusion process is to convert a set of samples from either Gaussian (for numerical variables) or uniform distributions (for categorical variables) into clean samples that represent a complex real-world distribution. The sampling proceeds iteratively, moving from timestep $t = T$, where the distribution is simple and noisy, to timestep $t = 0$, where the representation becomes clean and realistic.

The iterative reverse diffusion process for the generation of one sample is described in Algorithm 1. This algorithm requires four inputs: the denoising function, the conditional mask, the conditioning values, and the number of timesteps. The denoising function, implemented as a neural network, removes noise from the data and is shared across all variables to enhance its capacity

Algorithm 2: Training Process

Inputs: Denoising function dn_θ, number of Timesteps T, original data x_0^n, x_0^c
Initialize the weights θ of the Denoising function dn_θ.
Preprocess the original data x_0^n, x_0^c
for iteration=1, 2, ..., N **do**
  Sample a batch of original features, $x_0^n,\ x_0^c$.
  Sample a batch of timesteps t uniformly distributed within the interval $[0, T]$.
  for each numerical features x_0^n **do**
    Use the forward Gaussian diffusion process of Eq. 5 (with reparametrization trick) to obtain $x_t^{j_n}$ and ϵ_t^n.
  end for
  for each categorical feature x_0^c **do**
    Use the forward multinomial diffusion process of Eq. 9 to obtain x_t^c.
  end for
  Generate the new conditional Mask
  Compute the denoising function outputs: $\hat{\epsilon}_t^n, \hat{x}_0^c = dn_\theta\left(x_t^n, x_t^c, \texttt{Mask}, x_0^n, x_0^c, t\right)$.
  Use Eq. 8 to compute L_{simple} (Mask) for numerical features with Mask = 1; using ϵ_t^n and $\hat{\epsilon}_t^n$
  Use Eq. 1 to compute L_t (Mask) for categorical features with Mask = 1; using x_0^c, $\hat{x}_0^c$ and x_t^c
  Compute the total loss function of Eq. 11.
  Minimize the total loss function and update the denoising function weights.
end for

to learn complex relationships. The conditional mask distinguishes the conditioning variables, x^{cond} and y, from those involved in the forward diffusion process that require denoising. Variables requiring denoising are marked with Mask = 1, while those that do not are marked with Mask = 0. The conditioning values are given by x^{cond} and y at time $t = 0$. The number of timesteps dictates how many reverse diffusion steps must be performed.

First, a consistent preprocessing procedure is applied to the conditioning values, using the Gaussian quantile transformation from the scikit-learn library [43] for numerical features and ordinal encoding for categorical ones. Then, at time $t = T$, the process initiates by sampling the initial noise for each dataset feature separately: Gaussian noise for numerical features and uniform noise for categorical features. The iterative denoising process then proceeds from $t = T$ to $t = 0$. Each iteration involves three key actions: first, the denoising function dn_θ estimates $\hat{\epsilon}_t^n$ for continuous features and $\hat{x}_0^c$ for categorical features. Next, a reverse Gaussian step and a reverse multinomial step are performed using Equations (7) and (10) to estimate x_{t-1}^n and x_{t-1}^c, respectively. These actions are repeated iteratively until the final outputs x_0^n and x_0^c are reached. Finally, a postprocessing step is applied to the final outputs x_0^n and x_0^c to reverse the initial preprocessing. It is important to note that the preprocessing of conditioning values and postprocessing of outputs enhances the learning rate without impacting the quality of the final results. We tested various normalization methods for numerical features, such as Standard Scaler and Min-Max Scaler from scikit-learn, and found the results to be statistically equivalent.

4.4 Training Process

The training process for the denoising function is outlined in Algorithm 2. The primary aim is to train the neural network dn_θ used in the state 7 of Algorithm 1. This network is designed to leverage noisy representations of numerical and categorical features x_t^n, x_t^c, along with the conditional mask Mask, conditioning values, and current timestep t, to estimate both $\hat{\epsilon}_t^n$ and $\hat{x}_0^c$. For numerical features, $\hat{\epsilon}_t^n$ represents the estimated noise introduced during the forward diffusion process between timesteps $t-1$ and t. For categorical features, $\hat{x}_0^c$ serves as an estimate of the original categorical feature a $t = 0$.

The first steps involve initializing the model weights and preprocessing the original data as discussed in the previous section, followed by an iterative learning process. In each iteration, a batch of original features $x_0^n,\ x_0^c$ is sampled from the original dataset. For each element in the batch, a timestep t is then sampled uniformly from the interval $[0, T]$. Using the forward Gaussian diffusion process, targets for numerical features are computed. Specifically, for each numerical feature, the reparametrization trick is applied to Equation (5) to compute x_t^n, that is the noisy representation of the numerical feature at time t, and ϵ_t^n, the corresponding noise. Similarly, the forward multinomial diffusion process of Equation (9) is used to obtain x_t^c, the noisy representation of the categorical feature a time t. Next, a new mask is dynamically generated to select a subset of features that will serve as the conditioning input. With all the dn_θ inputs $(x_t^n, x_t^c, \texttt{Mask}, x_0^n, x_0^c, t)$ and targets (ϵ_t^n and x_0^c) in place, the loss function is then calculated.

The model is trained by minimizing the following total loss function:

$$L_t^{MTabGen} = \frac{L_t^{\text{simple}}(\texttt{Mask})}{K_{num}} + \frac{\sum_{i \le K_{cat}} L_t^i(\texttt{Mask})/Cl_i}{K_{cat}}. \tag{11}$$

The total loss function $L_t^{MTabGen}$ includes two main components:

- —Loss for *numerical features*: L_t^{simple} (defined in Equation (8)) computes the *MSE* between the true noise ϵ_t^n and the predicted noise $\hat{\epsilon}_t^n$ introduced in the forward Gaussian diffusion process between the steps $t-1$ and t.
- —Loss for *categorical features*: L_t^i (defined in Equation (1)) calculates the *KL* divergence between the posterior of the forward process $q\left(x_{t-1}^c | x_t^c, x_0^c\right)$ and parametrized reverse diffusion process $\hat{p}\left(x_{t-1}^c | x_t^c\right)$. Following the approach in [19], the $\hat{p}\left(x_{t-1}^c | x_t^c\right)$ is approximated by $q\left(x_{t-1}^c | x_t^c, \hat{x}_0^c\right)$ using Equation (10).

$L_t^{\text{simple}}(\texttt{Mask})$ and $L_t^i(\texttt{Mask})$ means that the loss functions are computed taking into account only the prediction error on variables affected by the forward diffusion process (i.e., $\texttt{Mask} = 1$). Here, Cl_i is the number of classes of the ith categorical variable.

4.5 Dynamic Conditioning

A key feature of the proposed solution is that the split between x^{mask} and x^{cond} does not have to be fixed for every row i in the dataset. The transformer encoder can manage mask with an arbitrary number of zeros/ones, so we can dynamically alter the split between x^{mask} and x^{cond} by just producing a new mask. In the extreme scenario, we can generate a new mask $\texttt{Mask}_i$ for each row i. During training, the number of ones in $\texttt{Mask}_i$ (i.e., the number of features to be included in the forward diffusion process) is uniformly sampled from the interval $[1, K_{num} + K_{cat}]$. A model that has been trained in this manner can then be used for both tasks, that is, generation of synthetic data ($\texttt{Mask}_i = 1$ for all the $K_{num} + K_{cat}$ features and for any dataset index i) and imputation of missing values (for each i, $\texttt{Mask}_i = 1$ for the feature to impute). As discussed, when the original dataset contains missing values, a new `MissingMask` is introduced and combined with the conditional `Mask` to remove any missing value from the condition. The `MissingMask` is fixed and constant during the training phase. This setup allows for more flexible conditioning scenarios.

Specifically:

- —When $x^{\text{cond}} = \emptyset$, $p\left(x_{t-1}^{\text{mask}} | x_t^{\text{mask}}, x^{\text{cond}}, y\right) = p\left(x_{t-1} | x_t, y\right)$ our model aligns with *TabDDPM*, generating synthetic data influenced by the target distribution.
- —When $\bar{x}^M \neq \emptyset$, the model can generate synthetic data based on the target distribution and either a fixed or dynamic subset of features. Conditioning on a fixed subset introduces advantages in settings where certain variables are readily accessible, whereas others are difficult to obtain

Table 1. Tabular Benchmark Properties

Dataset	Rows	Num. Feats	Cat. Feats	Task
HELOC	9,871	21	2	Binary
Churn	10,000	6	4	Binary
Gas Concentrations	13,910	129	0	Multiclass (6)
Cal. Hous.	20,640	8	0	Regression
House Sales	21,613	14	2	Regression
Adult Inc.	32,561	6	8	Binary
Otto Group	61,900	93	0	Multiclass (9)
Cardio	70,000	7	4	Binary
Insurance	79,900	8	2	Binary
Forest Cov.	581K	10	2	Multiclass (7)

due to challenges like cost constraints. In such cases, the scarce data can be synthetically produced using the known variables. Conversely, when conditioning on a dynamic subset of features, the model effectively addresses the challenge of imputing gaps within a dataset.

5 Experiments

5.1 Data

Below we introduce the benchmark datasets used in the performance evaluation of our model. The statistics are summarized in Table 1.

— ***Home Equity Line of Credit (HELOC)*** [13]: HELOC provided by FICO (a data analytics company) contains anonymized credit applications of HELOC credit lines. The dataset contains 21 numerical and 2 categorical features characterizing the applicant to the HELOC credit line. The task is a binary classification and the goal is to predict whether the applicant will make timely payments over a two-year period.
— *Churn Modelling* [21]: This dataset consists of six numerical and four categorical features about bank customers. The binary classification task involves predicting whether or not the customer closed their account.
— *Gas Concentrations* [58]: The dataset contains measurements from 16 chemical sensors exposed to 6 gases at different concentration levels. It contains 13.9M of rows and 129 continuous features and the classification task is to determine which is the gas generating the data.
— *California Housing* [42]: The information refers to the houses located in a certain California district, as well as some basic statistics about them based on 1990 census data. This is a regression task about forecasting the price of a property.
— *House Sales King Country* [25]: Similar to the California Housing case, this regression task involves estimating property prices in the King County region for sales between May 2014 and May 2015. The original dataset included 14 numerical features, 4 categorical features, and 1 date feature. During preprocessing, the date feature was transformed into two categorical variables: month and year.
— *Adult Incoming* [2]: Personal details such as age, gender, or education level are used to predict whether an individual would earn more or less than 50K$ per year.
— *Otto Group* [3]: This dataset, provided by the Otto Group (an e-commerce company), contains 61.9K of rows and 93 continuous product attributes. The task is a multiclass classification problem with nine categories, aiming to determine the category to which each product belongs.

— *Cardiovascular Disease* [54]: The existence or absence of cardiovascular disease must be predicted based on factual information, medical examination results, and information provided by the patient. The dataset consists of seven numerical and four categorical features.
— *Insurance* [46]: Customer variables and past payment data are used to solve a binary task: determining whether the customer will pay on time. The dataset has eight numerical and two categorical features.
— *Forest Cover Type* [6]: In this multiclass classification task with seven categories, cartographic variables are used to predict the forest cover type. The first eight features of the dataset are continuous, whereas the last two are categorical with 4 and 40 levels, respectively.

5.2 Baselines

For the synthetic data generation task, we consider the following state-of-the-art baselines drawn from representative generative modeling paradigms: VAE, GAN, and Diffusion Models:

— *TabDDPM* [31]: State-of-the-art diffusion model for tabular data generation and model in which we have premised the proposed approach.
— *Tabsyn*[1] [65]: Recent state-of-the-art tabular generative model that integrates a diffusion model into the continuous latent space projected by a VAE.
— *CoDi*[2] [33]: A diffusion model for tabular data generation. The *StaSy* and *CoDi* models are from the same team. In [33], the authors show that *CoDi* consistently outperforms *StaSy*. Therefore, we only include *CoDi* in our evaluation.
— *TVAE*[3] [62]: A variational autoencoder adapted for mixed-type tabular data.
— *CTGAN*[3] [62]: A conditional GAN for synthetic tabular data generator.

For the missing data imputation task, the following state-of-the-art baselines have been considered:

— *missForest*[4] [51]: Iterative method based on random forests to predict and fill in missing values.
— *GAIN*[4] [63]: GAN model for tabular missing value imputation.
— *HyperImpute*[4] [24]: Iterative imputation algorithm using both regression and classification methods based on linear models, trees, XGBoost, CatBoost, and neural nets.
— *Miracle*[4] [32]: Missing imputation algorithm using a causal deep learning approach.
— *TabCSDI*[5] [68]: State-of-the-art diffusion model for missing data imputation.

5.3 Metrics

We evaluate the generative models on three different dimensions: (1) ML efficiency, (2) statistical similarity, and (3) privacy risk.

5.3.1 ML Efficiency. The *ML efficiency* measures the performance degradation of classification or regression models trained on synthetic data, and then tested on real data. The basic idea is to use a ML discriminative model to evaluate the utility of synthetic data provided by a generative model. As demonstrated by Kotelnikov et al. [31], a strong ML model allows to obtain more stable and consistent conclusions on the performances of the generative model. Based on this intuition, we consider four different ML models: *XGBoost* [8], *CatBoost* [44], *LightGBM* [28], and *MLP*. We introduce an initial fine-tuning step, during which we derive the best hyperparameter

[1]GitHub: https://github.com/amazon-science/tabsyn.
[2]GitHub: https://github.com/ChaejeongLee/CoDi.
[3]We use the implementation provided by https://sdv.dev/.
[4]We use the implementation provided by https://github.com/vanderschaarlab/hyperimpute.
[5]GitHub: https://github.com/pfnet-research/TabCSDI.

configuration using Bayesian optimization and Optuna library [1]. Specifically, we perform 100 iterations to fine-tune the model's (*XGBoost*, *CatBoost*, *LightGBM*, and *MLP*) hyperparameters on each dataset's real data within the benchmark. Every hyperparameter configuration for ML model is cross-validated, using a five-fold split. The complete hyperparameter search space is shown in Appendix A (Table A1).

Once the discriminative model has been optimized for each dataset, the generative model is further cross-validated using a five-fold split, by implementing the following procedure. For each fold, the real data is split into three subsets. The main purpose of the first subset is to train the generative model. The resulting model generates a synthetic dataset conditioned on the second subset. The synthetic dataset is then used to train the discriminative model. The so-obtained ML model is finally tested on the third held-out subset, which has not been used in training any of the models. The procedure is repeated for each fold, and the obtained metric mean is used as a final measure to compute the generative model ML efficiency.

5.3.2 Statistical Similarity. The comparison between synthetic and real data accounts for both individual and joint feature distributions. Adopting the approach proposed by Zhao et al. [67], we employ Wasserstein [59] and Jensen-Shannon distances [34] to analyze numerical and categorical distributions. In addition, we use the square difference between pairwise correlation matrix to evaluate the preservation of feature interactions in synthetic datasets. Specifically, the Pearson correlation coefficient measures correlations between numerical features, the Theil uncertainty coefficient measures correlations between categorical features, and the correlation ratio evaluates interactions between numerical and categorical features.

5.3.3 Privacy Risk. The *Privacy Risk* is evaluated using the **Distance to Closest Record (DCR)**, i.e., the Euclidean distance between any synthetic record and its closest corresponding real neighbor. Ideally, the higher the DCR the lesser the risk of privacy breach. It is important to note that out-of-distribution data, i.e., random noise, will also produce high DCR. Therefore, to maintain ecological validity, the DCR metric needs to be evaluated jointly with the ML efficiency metric.

6 Results

6.1 ML Efficiency

6.1.1 Synthetic Data Generation. In this task, we evaluate the performance of our generative model in producing high-quality synthetic data, conditioned exclusively by the supervised target y. To this end, we consider two variants of our model:

(1) *MTabGen I*: This variant consistently includes all dataset features in the diffusion process and was specifically designed for the synthetic data generation task.
(2) *MTabGen II*: During training, this variant dynamically selects which features are incorporated in the diffusion process, making it versatile for both imputing missing data and generating complete synthetic datasets.

The results shown in Table 2 indicate that *MTabGen II* demonstrates competitive performance compared to existing state-of-the-art methods like *TabDDPM*, *Tabsyn*, *CoDi*, *TVAE*, or *CTGAN* in the synthetic data generation task, while showing moderate but statistically significant[6] improvements in most of the datasets. However, the specialized *MTabGen I* achieves the best performance across

[6]Following the recommendation of [45], we applied the Wilcoxon signed-rank test to compare, for each dataset, the results obtained by the best-performing *MTabGen* model against the best-performing baseline. In 6 out of 10 cases-specifically, HELOC, California Housing, House Sales, Adult Income, Otto, and Forest Cover Type, *MTabGen* demonstrated a statistically significant improvement over the strongest baseline, with a p-value < 0.01.

Table 2. ML Efficiency

Dataset	Baseline	TVAE	CTGAN	CoDi	Tabsyn	TabDDPM	MTabGen I	MTabGen II
HELOC ↑	83.69 ± 0.04	79.41 ± 0.05	77.53 ± 0.06	75.82 ± 0.07	79.24 ± 0.05	76.69 ± 0.10	**82.91 ± 0.07**	82.71 ± 0.08
Churn ↑	85.25 ± 0.05	81.67 ± 0.08	79.31 ± 0.06	82.77 ± 0.15	**84.60 ± 0.06**	83.62 ± 0.15	84.43 ± 0.03	83.91 ± 0.04
Gas ↑	99.47 ± 0.09	94.55 ± 0.07	62.04 ± 0.07	65.41 ± 0.04	98.66 ± 0.04	65.51 ± 0.06	**98.80 ± 0.07**	98.60 ± 0.06
Cal. Hous. ↓	0.161 ± 0.001	0.316 ± 0.003	0.488 ± 0.004	0.290 ± 0.003	0.256 ± 0.002	0.272 ± 0.002	**0.224 ± 0.001**	0.227 ± 0.001
House Sales ↓	0.101 ± 0.001	0.209 ± 0.001	0.335 ± 0.001	0.159 ± 0.001	0.148 ± 0.001	0.145 ± 0.001	**0.121 ± 0.001**	0.145 ± 0.001
Adult Inc. ↑	86.95 ± 0.04	84.34 ± 0.07	83.64 ± 0.08	84.43 ± 0.08	84.70 ± 0.06	84.86 ± 0.07	**85.30 ± 0.09**	85.15 ± 0.07
Otto ↑	81.50 ± 0.06	63.87 ± 0.05	50.48 ± 0.05	63.21 ± 0.06	67.14 ± 0.08	63.34 ± 0.08	**73.04 ± 0.07**	72.98 ± 0.06
Cardio ↑	73.54 ± 0.05	72.47 ± 0.08	71.81 ± 0.06	72.34 ± 0.10	72.90 ± 0.11	72.88 ± 0.14	**72.97 ± 0.08**	72.67 ± 0.12
Insurance ↑	92.00 ± 0.03	92.63 ± 0.11	92.56 ± 0.05	92.03 ± 0.07	92.20 ± 0.05	92.21 ± 0.04	**92.77 ± 0.11**	92.64 ± 0.07
Forest Cov. ↑	96.42 ± 0.06	70.36 ± 0.04	65.65 ± 0.07	74.64 ± 0.09	74.83 ± 0.10	82.08 ± 0.07	**85.61 ± 0.04**	84.32 ± 0.09
Average Rank		4.9	6.5	5.7	3.3	4.0	**1.1**	2.4

Classification tasks use F1-score, and regression tasks use MSE, indicated by up/down arrows for maximization/minimization of the metric. Cross-validation mean and standard deviation are shown for each dataset-model pair. Best and second-best results are highlighted in bold and underline, respectively. *Baseline* column shows average performance of ML models trained on real data, while other columns reflect average performance of ML models trained on synthetic data from specified models. All models are tested on real data.

the evaluated tasks. The key outcomes of our experiments are as follows: (1) *TVAE* produces better results than *CTGAN*. (2) Approaches based on Diffusion Models outperform *TVAE* and *CTGAN* on average. (3) Our two proposed models show overall improved performance over *TabDDPM*, *Tabsyn*, and *CoDi*, with statistically significant improvements in specific datasets.[6] In the remaining cases, the results were on par with the baselines, without a statistically significant difference. (4) Our model tends to outperform the baselines in datasets with a large number of features, such as the *Gas Concentrations* and *Otto Group* datasets, although this trend is less consistent with *Tabsyn*. Additionally, it is worth noting that the ML efficiency results align with those reported in the *Tabsyn* paper, where *Tabsyn* generally surpasses *TabDDPM* and *CoDi*, with *TVAE* also showing stronger performance than *CoDi*.

The results presented in Table 2 are obtained after applying Bayesian optimization for each generative model, using the Optuna library over 100 trials, and evaluating performance with the cross-validated ML efficiency metric defined in Section 5.3 as the objective. A similar hyperparameter optimization procedure was applied to all baseline models, considering the parameters specific to each method, enabling fair comparison with *MTabGen*. The specific hyperparameter search space for each model is shown in Appendix A (Table A2).

6.1.2 Ablation Study. We conduct an ablation study to evaluate the contributions of the encoder–decoder transformer and dynamic conditioning to our model's performance and usability. The encoder–decoder transformer enhances performance for two primary reasons:

— *Enhanced Learning of Inter-Feature Interactions within Condition (Encoder) and Masked Features (Decoder)*: Transformer layers allow for better learning of inter-feature interactions compared to MLPs. This is primarily due to the attention mechanism, where the new embedding of a feature x_i is computed by a linear combination of the embeddings of all features $\{x_j\}$. The weight of feature x_j depends on the current values of x_i and x_j. This mechanism is more flexible than in MLP case, where the contribution of feature x_j to a neuron in the next layer is fixed and does not depend on its current value.

— *Conditioning Attention Mechanism*: Our model, *MTabGen*, uses an encoder–decoder transformer architecture. The encoder learns latent representations of unmasked features for conditioning, while the decoder focuses on learning latent representations of masked or

Table 3. *Ablation Study* in Terms of *ML Efficiency*

Dataset	Baseline	TabDDPM	TabDDPM-Transf	MTabGen I
HELOC ↑	83.69 ± 0.04	76.69 ± 0.10	81.15 ± 0.08	**82.91 ± 0.07**
Churn ↑	85.25 ± 0.05	83.62 ± 0.15	83.62 ± 0.05	**84.43 ± 0.03**
Gas ↑	99.47 ± 0.09	65.51 ± 0.06	93.35 ± 0.09	**98.80 ± 0.09**
Cal. Hous. ↓	0.161 ± 0.001	0.272 ± 0.001	0.232 ± 0.001	**0.224 ± 0.001**
House Sales ↓	0.101 ± 0.001	0.145 ± 0.001	0.128 ± 0.001	**0.121 ± 0.001**
Adult Inc. ↑	86.95 ± 0.04	84.86 ± 0.07	85.13 ± 0.06	**85.30 ± 0.09**
Otto ↑	81.50 ± 0.06	63.34 ± 0.08	71.01 ± 0.08	**73.04 ± 0.07**
Cardio ↑	73.54 ± 0.05	72.88 ± 0.14	72.94 ± 0.11	**72.97 ± 0.08**
Insurance ↑	92.00 ± 0.03	92.21 ± 0.04	92.69 ± 0.06	**92.77 ± 0.11**
Forest Cov. ↑	96.42 ± 0.06	82.08 ± 0.07	83.84 ± 0.08	**85.61 ± 0.04**

Classification tasks use F1-score, and regression tasks use MSE, indicated by up/down arrows for maximization/minimization of the metric. Cross-validation mean and standard deviation are shown for each dataset-model pair. Best and second-best results are highlighted in bold and underline, respectively. *Baseline* column shows average performance of ML models trained on real data, while other columns reflect average performance of ML models trained on synthetic data from specified models. All models are tested on real data.

noisy features. By incorporating conditioning within the attention mechanism of the transformer decoder, we reduce learning bias compared to conventional methods like those used in *TabDDPM*, which simply sum the latent representations of conditions and noisy features.

To test the impact of these hypotheses, we modified the original implementation of *TabDDPM* by replacing the MLP denoising model with a transformer encoder, while retaining the same conditioning mechanism (i.e., summing the condition embedding and feature embedding). We call this new implementation *TabDDPM-Transf*. As shown in the first two columns of Table 3, our transformer-enhanced *TabDDPM-Transf* consistently outperforms the standard *TabDDPM*. The impact of the conditioning attention mechanism is further demonstrated by the comparison of *TabDDPM-Transf* and *MTabGen I* in Table 3.

With respect to model usability, dynamic conditioning allows a single model to handle a variety of tasks without compromising performance. Notably, performance metrics for our multitasking model, *MTabGen II* (with dynamic conditioning), align closely with those of our specific data generation model, *MTabGen I* (without dynamic conditioning), as shown in Table 2. Dynamic conditioning supports not only synthetic data generation and missing data imputation but also "prompted data generation"—a scenario where a synthetic subset of features is generated based on a known subset of variables. This method is particularly useful in settings where data collection is challenging, enhancing data augmentation in ML projects, improving customer profiling, or acting as a simulated environment in reinforcement learning, which accelerates data-generation efforts.

6.1.3 Missing Data Imputation. Here, we report the results of our experiments on evaluating the models' ability to impute missing values. Specifically, in this task, the generative model utilizes the available data to condition the generation of data for the missing entries.

The evaluation of imputed data quality considers three possible scenarios for missing data: **Missing Completely at Random (MCAR)**, **Missing at Random (MAR)**, and MNAR. For each scenario, we first divide the dataset into three subsets: 40% for "imputation training," 30% for "imputation testing/discriminative training," and 30% as a hold-out set. Next, we generate three versions of the imputation training and imputation test/discriminative train splits, each containing an increasing proportion of missing data (10%, 25%, and 40%), while keeping the hold-out set intact.

Algorithm 3: Evaluate Imputation Model Performance

```
for each dataset in the benchmark do
    for each missing value type {MCAR, MAR, MNAR} do
        for each percentage of missing values in {10%, 25%, 40%} do
            Train the generative model on 30% "imputation training" subset.
            Impute the missing values in the 30% "imputation testing/discriminative training" subset.
            Train the ML discriminative model (e.g., XGBoost, CatBoost, LightGBM and MLP) on the imputed data.
            Evaluate the ML discriminative model on the hold-out 30% subset.
        end for
    end for
end for
```

Table 4. *Missing Imputation* Analysis in Terms of *ML Efficiency* When the Missing Information is MCAR

Dataset	% Missing	missForest	GAIN	HyperImpute	Miracle	TabCSDI	MTabGen
HELOC ↑	10%	83.16 ± 0.02	82.44 ± 0.02	83.36 ± 0.04	83.39 ± 0.03	83.30 ± 0.05	**83.79 ± 0.03**
	25%	82.80 ± 0.04	81.92 ± 0.04	83.23 ± 0.04	82.85 ± 0.04	83.27 ± 0.04	**83.54 ± 0.04**
	40%	82.59 ± 0.04	81.74 ± 0.03	82.97 ± 0.03	82.74 ± 0.03	82.67 ± 0.03	**83.41 ± 0.04**
Churn ↑	10%	84.49 ± 0.03	84.66 ± 0.03	84.47 ± 0.05	84.60 ± 0.03	84.48 ± 0.03	**84.76 ± 0.03**
	25%	84.13 ± 0.03	83.45 ± 0.05	83.89 ± 0.05	84.08 ± 0.03	83.92 ± 0.03	**84.51 ± 0.04**
	40%	83.08 ± 0.04	83.22 ± 0.04	83.14 ± 0.04	82.52 ± 0.04	83.45 ± 0.04	**84.18 ± 0.04**
Cal. Hous. ↓	10%	0.182 ± 0.002	0.182 ± 0.002	0.173 ± 0.002	0.174 ± 0.002	0.175 ± 0.002	**0.170 ± 0.002**
	25%	0.229 ± 0.002	0.249 ± 0.002	0.190 ± 0.002	0.196 ± 0.002	0.188 ± 0.003	**0.183 ± 0.002**
	40%	0.269 ± 0.004	0.306 ± 0.004	0.218 ± 0.003	0.269 ± 0.004	0.258 ± 0.004	**0.210 ± 0.003**
House Sales ↓	10%	0.109 ± 0.002	0.110 ± 0.002	0.112 ± 0.002	0.106 ± 0.002	0.110 ± 0.002	**0.105 ± 0.002**
	25%	0.118 ± 0.002	0.131 ± 0.003	0.120 ± 0.003	0.127 ± 0.002	0.117 ± 0.003	**0.115 ± 0.002**
	40%	0.155 ± 0.003	0.145 ± 0.003	0.156 ± 0.003	0.130 ± 0.003	0.128 ± 0.003	**0.119 ± 0.002**

Classification tasks use F1-score, and regression tasks use MSE, indicated by up/down arrows for maximization/minimization of the metric. Cross-validation mean and standard deviation are shown for each dataset-model pair. Best and second-best results are highlighted in bold and underline, respectively.

To assess the quality of the imputed data in terms of ML efficiency, we follow the steps outlined in Algorithm 3. For the MAR and MNAR scenarios, the missing data are generated using the code used by [24] and [38].

The results of the experiments for the MCAR, MAR, and MNAR scenarios are presented in Tables 4, 5, and 6, respectively. As the assumption about missingness becomes more complex (MCAR → MAR → MNAR), all the models included in the comparison exhibit a very slight degradation in performance. However, the findings remain consistent across all missing data scenarios.

Our experiments indicate that *MTabGen* tends to outperform the baseline models considered in this evaluation: *missForest* [52], *GAIN* [63], *HyperImpute* [24], *Miracle* [32], and *TabCSDI* [68]. Across all levels of missing data, *MTabGen* generally achieves better performance on average, with its advantage becoming more pronounced as the proportion of missing data increases.

6.2 Statistical Similarity

In this task, we compare synthetic and real data accounts based on individual and joint feature distributions. We report only the top three models with the best ML efficiency. This approach is motivated by our findings, which show a strong association between the ML efficiency of synthetic

Table 5. *Missing Imputation* Analysis in Terms of *ML Efficiency* When the Missing Information is MAR.

Dataset	% Missing	missForest	GAIN	HyperImpute	Miracle	TabCSDI	MTabGen
HELOC ↑	10%	83.13 ± 0.02	82.39 ± 0.04	83.34 ± 0.02	83.30 ± 0.03	83.25 ± 0.03	**83.78 ± 0.02**
	25%	82.75 ± 0.03	81.88 ± 0.02	83.15 ± 0.03	82.81 ± 0.02	83.18 ± 0.01	**83.53 ± 0.04**
	40%	82.42 ± 0.02	81.64 ± 0.04	82.86 ± 0.04	82.52 ± 0.02	82.59 ± 0.05	**83.40 ± 0.04**
Churn ↑	10%	84.46 ± 0.04	84.55 ± 0.03	84.42 ± 0.02	84.58 ± 0.04	84.44 ± 0.03	**84.76 ± 0.03**
	25%	84.05 ± 0.03	83.39 ± 0.02	83.79 ± 0.02	84.00 ± 0.02	83.58 ± 0.02	**84.50 ± 0.02**
	40%	82.91 ± 0.05	83.11 ± 0.01	83.00 ± 0.03	82.40 ± 0.04	83.31 ± 0.03	**84.17 ± 0.04**
Cal. Hous. ↓	10%	0.182 ± 0.003	0.183 ± 0.004	0.173 ± 0.003	0.174 ± 0.003	0.175 ± 0.004	**0.171 ± 0.003**
	25%	0.230 ± 0.002	0.251 ± 0.004	0.191 ± 0.002	0.197 ± 0.004	0.190 ± 0.005	**0.183 ± 0.002**
	40%	0.272 ± 0.002	0.308 ± 0.005	0.222 ± 0.003	0.274 ± 0.002	0.264 ± 0.004	**0.211 ± 0.003**
House Sales ↓	10%	0.110 ± 0.004	0.112 ± 0.002	0.114 ± 0.004	0.106 ± 0.002	0.111 ± 0.004	**0.105 ± 0.002**
	25%	0.120 ± 0.003	0.134 ± 0.002	0.122 ± 0.004	0.130 ± 0.002	0.121 ± 0.003	**0.116 ± 0.003**
	40%	0.158 ± 0.003	0.149 ± 0.001	0.160 ± 0.003	0.134 ± 0.003	0.132 ± 0.003	**0.120 ± 0.002**

Classification tasks use F1-score, and regression tasks use MSE, indicated by up/down arrows for maximization/minimization of the metric. Cross-validation mean and standard deviation are shown for each dataset-model pair. Best and second-best results are highlighted in bold and underline, respectively.

Table 6. *Missing Imputation* Analysis in Terms of *ML Efficiency* When the Missing Information is MNAR

Dataset	% Missing	missForest	GAIN	HyperImpute	Miracle	TabCSDI	MTabGen
HELOC ↑	10%	83.09 ± 0.03	82.35 ± 0.03	83.31 ± 0.04	83.25 ± 0.04	83.23 ± 0.04	**83.75 ± 0.04**
	25%	82.59 ± 0.03	81.78 ± 0.02	83.04 ± 0.05	82.69 ± 0.05	83.07 ± 0.05	**83.47 ± 0.03**
	40%	82.12 ± 0.04	81.43 ± 0.03	82.67 ± 0.03	82.35 ± 0.03	82.31 ± 0.04	**83.31 ± 0.04**
Churn ↑	10%	84.42 ± 0.04	84.51 ± 0.05	84.36 ± 0.04	84.49 ± 0.03	84.38 ± 0.04	**84.72 ± 0.04**
	25%	83.95 ± 0.04	83.27 ± 0.04	83.66 ± 0.05	83.86 ± 0.04	83.72 ± 0.03	**84.42 ± 0.03**
	40%	82.72 ± 0.03	82.93 ± 0.05	82.80 ± 0.05	82.23 ± 0.03	83.09 ± 0.04	**84.04 ± 0.05**
Cal. Hous. ↓	10%	0.184 ± 0.003	0.185 ± 0.004	0.177 ± 0.004	0.178 ± 0.003	0.176 ± 0.003	**0.173 ± 0.003**
	25%	0.234 ± 0.004	0.255 ± 0.003	0.200 ± 0.003	0.205 ± 0.002	0.194 ± 0.004	**0.187 ± 0.003**
	40%	0.285 ± 0.003	0.318 ± 0.002	0.239 ± 0.004	0.289 ± 0.003	0.279 ± 0.002	**0.220 ± 0.002**
House Sales ↓	10%	0.111 ± 0.002	0.116 ± 0.004	0.117 ± 0.004	0.112 ± 0.002	0.115 ± 0.003	**0.108 ± 0.003**
	25%	0.127 ± 0.001	0.140 ± 0.002	0.129 ± 0.003	0.137 ± 0.004	0.126 ± 0.002	**0.120 ± 0.002**
	40%	0.169 ± 0.002	0.160 ± 0.003	0.171 ± 0.003	0.143 ± 0.003	0.140 ± 0.004	**0.126 ± 0.003**

Classification tasks use F1-score, and regression tasks use MSE, indicated by up/down arrows for maximization/minimization of the metric. Cross-validation mean and standard deviation are shown for each dataset-model pair. Best and second-best results are highlighted in bold and underline, respectively

data and their ability to replicate the statistical properties of real data. In other words, synthetic data with higher ML utility tends to better reproduce both individual and joint feature distributions.

Table 7(a) shows the average Wasserstein Distance between synthetic and real numerical data distributions. Specifically, the Wasserstein distance is calculated for each numerical column between the real data and the synthetic data generated by *Tabsys*, *TabDDPM*, and *MTabGen*. For each generative model, the final dataset results are the average of these distances across all numerical columns in the dataset. *MTabGen* consistently performs better than *Tabsys* and *TabDDPM*. This advantage is more pronounced in datasets where there is a larger difference in ML efficiency, such as *HELOC*, *California Housing*, *House Sales*, or *Forest Cover Type* dataset. In contrast, in datasets like *Insurance*, where all models have similar ML efficiency, the Wasserstein distances are also

Table 7. Statistical Similarity Details

(a) Average Wasserstein Distance

Dataset	Tabsyn	TabDDPM	MTabGen
HELOC	0.25 ± 0.03	0.29 ± 0.02	**0.15 ± 0.02**
Churn	**0.15 ± 0.02**	0.23 ± 0.02	0.17 ± 0.02
Gas	**0.21 ± 0.04**	0.65 ± 0.06	0.23 ± 0.02
Cal. Hous	0.31 ± 0.03	0.34 ± 0.04	**0.22 ± 0.02**
House Sales	0.32 ± 0.02	0.28 ± 0.03	**0.15 ± 0.01**
Adult Inc.	**0.18 ± 0.03**	0.24 ± 0.02	0.19 ± 0.02
Otto	0.25 ± 0.02	0.27 ± 0.03	**0.16 ± 0.02**
Cardio	**0.23 ± 0.04**	0.26 ± 0.03	0.24 ± 0.03
Insurance	**0.19 ± 0.02**	0.22 ± 0.02	0.20 ± 0.03
Forest Cov.	0.41 ± 0.05	0.35 ± 0.04	**0.18 ± 0.02**
Av. Rank	1.7	2.8	**1.5**

(b) Average Jensen-Shannon Distance

Dataset	Tabsyn	TabDDPM	MTabGen
HELOC	0.15 ± 0.02	0.20 ± 0.02	**0.08 ± 0.01**
Churn	**0.07 ± 0.01**	0.15 ± 0.02	0.09 ± 0.01
Gas	NA	NA	NA
Cal. Hous	NA	NA	NA
House Sales	0.22 ± 0.02	0.26 ± 0.03	**0.09 ± 0.01**
Adult Inc.	**0.07 ± 0.01**	0.14 ± 0.02	0.08 ± 0.02
Otto	NA	NA	NA
Cardio	**0.10 ± 0.01**	0.14 ± 0.02	0.11 ± 0.01
Insurance	**0.09 ± 0.01**	0.13 ± 0.01	0.10 ± 0.01
Forest Cov.	0.24 ± 0.03	0.16 ± 0.02	**0.08 ± 0.01**
Av. Rank	**1.6**	2.9	**1.6**

(c) Average L2 Dist. Correlation Matrix

Dataset	Tabsyn	TabDDPM	MTabGen
HELOC	0.036 ± 0.004	0.040 ± 0.004	**0.025 ± 0.003**
Churn	**0.027 ± 0.003**	0.034 ± 0.003	0.028 ± 0.003
Gas	**0.032 ± 0.003**	0.078 ± 0.007	0.034 ± 0.004
Cal. Hous	0.042 ± 0.004	0.045 ± 0.005	**0.031 ± 0.003**
House Sales	0.044 ± 0.004	0.039 ± 0.004	**0.024 ± 0.003**
Adult Inc.	**0.029 ± 0.003**	0.036 ± 0.004	0.030 ± 0.003
Otto	0.037 ± 0.004	0.041 ± 0.004	**0.026 ± 0.003**
Cardio	**0.032 ± 0.003**	0.035 ± 0.003	0.033 ± 0.003
Insurance	0.031 ± 0.003	0.033 ± 0.003	**0.029 ± 0.003**
Forest Cov.	0.052 ± 0.004	0.046 ± 0.004	**0.031 ± 0.003**
Av. Rank	1.8	2.8	**1.4**

comparable. A qualitative analysis of the results for some of the columns can be found in the four plots in the first two rows plots of Figure 4. These plots compare the distributions of real data with those of the synthetic data generated by the different models.

Similarly, Table 7(b) shows the average Jensen-Shannon distance between synthetic and real categorical data distributions. *MTabGen* and *Tabsyn* outperform *TabDDPM*, consistent with the findings of [65], where *Tabsyn* achieved better results than *TabDDPM* on categorical data. Also the Jensen-Shannon distance appears to be strongly related to ML efficiency. For example, in datasets like *HELOC*, *House Sales* or *Forest Cover Type*, where ML efficiency of *MTabGen* is significantly better than the other baselines, it also has a lower Jensen-Shannon distance. In the *Insurance* dataset, where all baselines have similar ML efficiency, the Jensen-Shannon distances are also similar. Notably, while *MTabGen* and *Tabsyn* show similar overall results, a qualitative analysis of per-column results reveals that *MTabGen* tends to excel when the number of classes in the categorical variable increases, as shown in the last two rows plots of Figure 4.

Table 7(c) shows the average L2 distance between the two correlation matrices computed on real and synthetic data. Figure 5, instead, shows the L2 distance details across all the datasets in the benchmark. In Figure 5, more intense green color means higher difference between the real and synthetic correlation values. Even more than in the previous statistical measures, there exists an association between ML efficiency and L2 distance between correlation matrices. In datasets like *HELOC*, *California Housing*, *House Sales*, or *Forest Cover Type* where the ML efficiency of *MTabGen* is notably better than the one of *Tabsyn* and *TabDDPM* the corresponding heatmap of *MTabGen* are more lighter, e.g., they show a smaller error in the correlation estimation. Nevertheless, in datasets like *Cardio* or *Insurance* where all the models perform similarly, the heatmaps do not show resignable differences.

6.3 Privacy Risk

In this section, we delve deeper into the privacy risk associated with synthetic data. As previously mentioned, DCR is defined by the Euclidean distance between any synthetic record and its closest real neighbor. Ideally, a higher DCR indicates a lower privacy risk. However, out-of-distribution data (random noise) can also result in high DCR. Therefore, to maintain ecological validity, DCR should be evaluated alongside the ML efficiency metric. For this reason, our privacy risk evaluation includes only *Tabsyn*, *TabDDPM*, and *MTabGen*. These models have superior ML efficiency and are more likely to pose a privacy risk (lower DCR) because they closely mimic the original data.

Table 8 presents our evaluation results regarding the privacy risk metric, emphasizing the tradeoff between ML efficiency and privacy guarantees. Based on a 5% threshold for relative improvement in ML efficiency of *MTabGen* over *Tabsyn* and *TabDDPM*, the results are categorized to illustrate two key points: (1) Improvements in ML efficiency correlate with increased privacy risks. (2) When

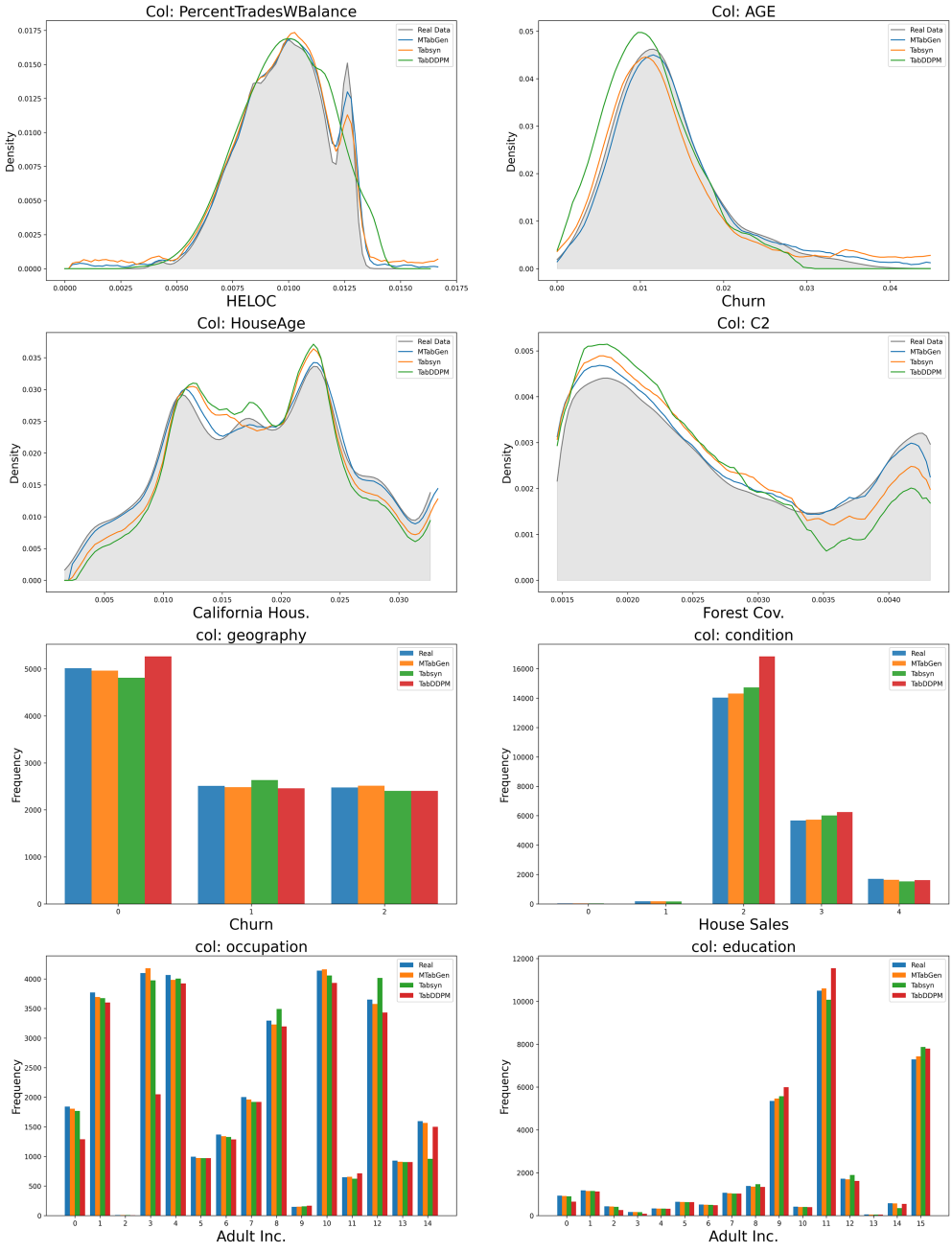

Fig. 4. Comparison between distribution of real and synthetic data. The two top rows present four examples of numerical columns from various datasets in our benchmark, while the two bottom rows contain examples of categorical features.

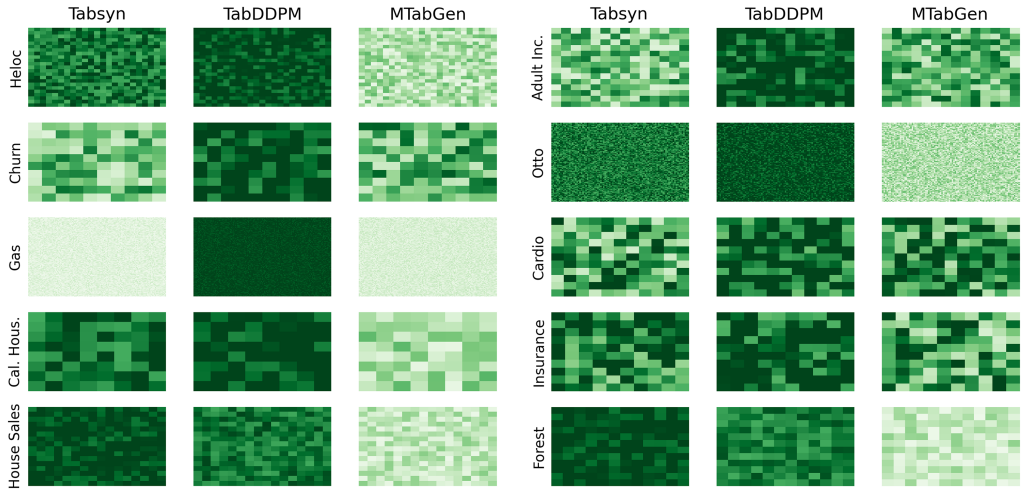

Fig. 5. L2 distance between correlation matrices computed on real and synthetic data. More intense green color means higher difference between the real and synthetic correlation values.

Table 8. Comparison of Privacy Risk (↑) and ML Efficiency (↑ for F1-Score, ↓ for MSE) for *Tabsyn*, *TabDDPM*, and *MTabGen* for Each Dataset

Tabsyn and MTabGen

Dataset	Tabsyn		MTabGen	
	Risk ↑	ML Eff.	Risk ↑	ML Eff.
HELOC	**0.35**	79.24 ↑	0.25	**82.91** ↑
Churn	0.06	**84.60** ↑	**0.09**	84.43 ↑
Gas	**0.21**	98.66 ↑	0.19	**98.80** ↑
Adult Inc.	**0.13**	84.70 ↑	0.11	**85.30** ↑
Cardio	0.35	72.90 ↑	**0.41**	**72.97** ↑
Insurance	**0.12**	92.20 ↑	**0.12**	**92.77** ↑
5% Relative ML Eff. Threshold				
Cal. Hous.	**0.81**	0.256 ↓	0.18	**0.224** ↓
House Sales	**0.16**	0.148 ↓	0.10	**0.121** ↓
Otto	**1.08**	67.14 ↑	0.12	**73.04** ↑
Forest Cov.	**0.85**	74.83 ↑	0.35	**85.61** ↑

TabDDPM and MTabGen

Dataset	TabDDPM		MTabGen	
	Risk ↑	ML Eff.	Risk ↑	ML Eff.
Churn	**0.09**	83.62 ↑	**0.09**	**84.43** ↑
Adult Inc.	**0.15**	84.86 ↑	0.11	**85.30** ↑
Cardio	**0.41**	72.88 ↑	**0.41**	**72.97** ↑
Insurance	**0.12**	92.21 ↑	**0.12**	**92.77** ↑
Forest Cov.	**0.47**	82.08 ↑	0.35	**85.61** ↑
5% Relative ML Eff. Threshold				
HELOC	**2.75**	76.69 ↑	0.25	**82.91** ↑
Gas	**3.15**	65.51 ↑	0.19	**98.80** ↑
Cal. Hous.	**1.28**	0.272 ↓	0.18	**0.224** ↓
House Sales	**0.34**	0.145 ↓	0.10	**0.121** ↓
Otto	**2.57**	63.34 ↑	0.12	**73.04** ↑

Privacy risk is evaluated using the DCR; higher values indicate a lesser risk of privacy breach. The results are divided based on a 5% ML efficiency threshold relative to the performance improvements of *MTabGen* compared to *Tabsyn* and *TabDDPM*. In cases where *MTabGen* achieves comparable ML efficiency to the baselines, the privacy risk is similar. However, in scenarios where *MTabGen* outperforms the other baselines, there is higher privacy risk (i.e., lower value of DCR distance), as expected, since better ML efficiency is related with closer statistical fidelity of the generated data.

MTabGen's performance improvement over the baselines is less than 5%, no significant increase in privacy risk is observed. However, when *MTabGen*'s performance is substantially better, the comparison becomes irrelevant because the synthetic data generated by the baselines deviate significantly from real data in terms of ML utility and statistical properties, resulting in a high DCR due to poor alignment with real data properties.

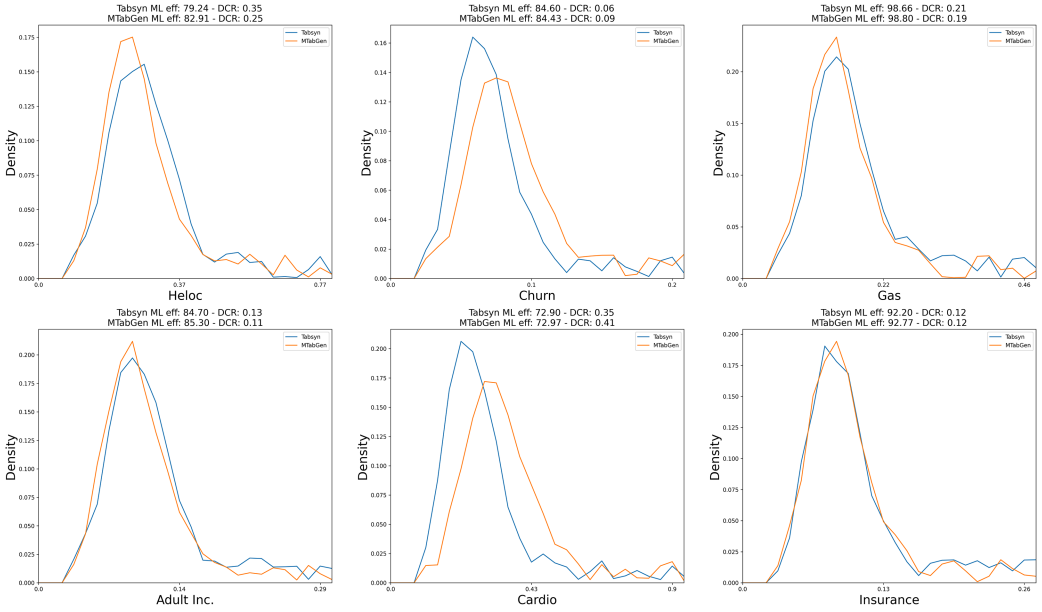

Fig. 6. DCR for *Tabsyn* and *MTabGen*.

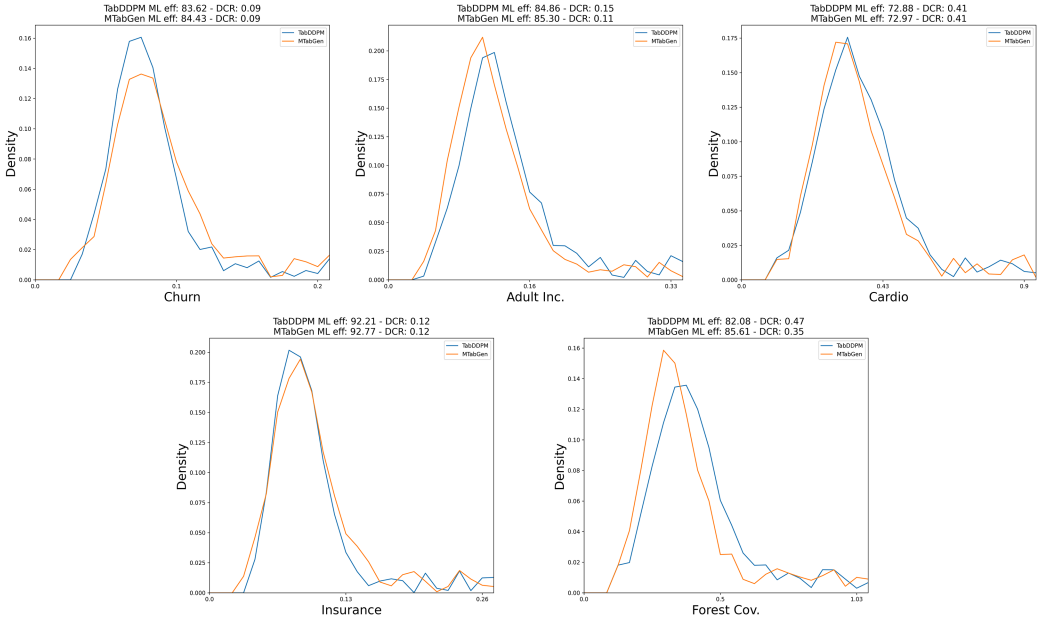

Fig. 7. DCR for *TabDDPM* and *MTabGen*.

Figures 6 and 7 provide a qualitative comparison of the DCR distribution for *MTabGen* versus *Tabsyn* and *TabDDPM* when the relative improvement in ML efficiency is less than 5%. Specifically, for each synthetic data point, we compute the distance to its closest record in the training dataset. The plots then depict the distribution of these distances. This comparison confirms that there are

no significant changes or patterns, indicating that the improvement in ML efficiency does not lead to a noticeable increase in privacy risk.

To mitigate the risk of increasing privacy concerns, our framework is equipped to integrate additional privacy-preserving measures, such as differential privacy [23], allowing for a better-controlled balance between data efficiency and privacy. Additionally, we conduct a sanity check to ensure that no synthetic sample perfectly matches any original sample (i.e., DCR is always greater than 0), safeguarding against direct data leakage.

7 Conclusion

In this article, we introduced *MTabGen*, a diffusion model enhanced with a conditioning attention mechanism, a transformer-based encoder–decoder architecture, and dynamic masking. *MTabGen* is specifically designed for applications involving mixed-type tabular data. The transformer encoder–decoder acts as the denoising network, enabling the conditioning attention mechanism while effectively capturing and representing complex interactions and dependencies within the data. The dynamic masking feature allows *MTabGen* to handle both synthetic data generation and missing data imputation tasks within a unified framework efficiently. We proposed to train the diffusion model to regenerate masked data, enabling applications ranging from data imputation to unconditioned or conditioned synthetic data generation. This versatility makes *MTabGen* suitable for generating synthetic data to overcome privacy regulations, augment existing datasets, or mitigate class imbalances. We evaluated *MTabGen* against established baselines across several public datasets with a diverse range of features. Our model demonstrated better overall performance in terms of ML efficiency and statistical accuracy, while maintaining privacy risks comparable to those of the baselines, particularly showing increased performance in datasets with a large number of features.

Impact Statements. Tabular data is one of the most common structures with which to represent information (e.g., finance, health). With the present model, the ability to generate or complete records in these structures is given. This fact requires ethical and privacy considerations. The authors encourage that before sharing any type of data, whether original or generated with the proposed model, to verify that reverse-identification is impossible or prevented by regulatory means. In addition to these considerations, our model and methods described in the article can also be utilized to rebalance datasets for minority groups by synthetically generating new samples conditioned on the minority class, thus aiding in fairer data representation. Apart from this, we see no other ethical issues related to this work.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '19)*. ACM, New York, NY, 2623–2631.

[2] Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20

[3] Wendy Kan Benjamin Bossan and Josef Feigl. 2015. Otto Group Product Classification Challenge. Retrieved from https://kaggle.com/competitions/otto-group-product-classification-challenge

[4] Dimitris Bertsimas, Colin Pawlowski, and Ying Daisy Zhuo. 2017. From predictive methods to missing data imputation: An optimization approach. *Journal of Machine Learning Research* 18, 1 (2017), 7133–7171.

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

[6] Jock Blackard. 1998. Covertype. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C50K5N

[7] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2024. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems* 35, 6 (2024), 7499–7519. DOI: https://doi.org/10.1109/TNNLS.2022.3229161

[8] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, 785–794.

[9] Bradley E. Cox, Kadian McIntosh, Robert D. Reason, and Patrick T. Terenzini. 2014. Working with missing data in higher education research: A primer and real-world example. *The Review of Higher Education* 37, 3 (2014), 377–402. DOI: https://doi.org/10.1353/rhe.2014.0026

[10] Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*. M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34, Curran Associates, Inc., 8780–8794. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf

[11] Cynthia Dwork. 2006. Differential privacy. In *Automata, Languages and Programming*. Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.), Springer, Berlin, 1–12.

[12] Justin Engelmann and Stefan Lessmann. 2021. Conditional Wasserstein GAN-based oversampling of tabular data for imbalanced learning. *Expert Systems with Applications* 174 (2021), 114582. DOI: https://doi.org/10.1016/j.eswa.2021.114582

[13] FICO. 2019. Home Equity Line of Credit (HELOC) Dataset. Retrieved from https://community.fico.com/s/explainable-machine-learning-challenge

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (Eds.), Curran Associates, Inc., Vol. 27. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2014/file/f033ed80deb0234979a61f95710dbe25-Paper.pdf

[15] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. 2024. Why do tree-based models still outperform deep learning on typical tabular data? In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22)*. Curran Associates Inc., Red Hook, NY, Article 37, 14 pages.

[16] Martijn W. Heymans and Jos W. R. Twisk. 2022. Handling missing data in clinical research. *Journal of Clinical Epidemiology* 151 (2022), 185–188. DOI: https://doi.org/10.1016/j.jclinepi.2022.08.016

[17] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep models under the GAN: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. ACM, New York, NY, 603–618. DOI: https://doi.org/10.1145/3133956.3134012

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*. H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33, Curran Associates, Inc., 6840–6851. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf

[19] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. 2021. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Advances in Neural Information Processing Systems*. M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34, Curran Associates, Inc., 12454–12465. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2021/file/67d96d458abdef21792e6d8e590244e7-Paper.pdf

[20] Niels Bruun Ipsen, Pierre-Alexandre Mattei, and Jes Frellsen. 2022. How to deal with missing data in supervised deep learning? In *Proceedings of the 10th International Conference on Learning Representations (ICLR '22)*.

[21] Shruti Iyyer. 2019. Churn Modelling. Kaggle. Retrieved from https://www.kaggle.com/datasets/shrutimechlearn/churn-modelling

[22] Sebastian Jäger, Arndt Allhorn, and Felix Bießmann. 2021. A benchmark for data imputation methods. *Frontiers in Big Data* 4 (2021), 693674. DOI: https://doi.org/10.3389/fdata.2021.693674

[23] Joonas Jälkö, Eemil Lagerspetz, Jari Haukka, Sasu Tarkoma, Antti Honkela, and Samuel Kaski. 2021. Privacy-preserving data sharing via probabilistic modeling. *Patterns* 2, 7 (2021), 100271.

[24] Daniel Jarrett, Bogdan C. Cebere, Tennison Liu, Alicia Curth, Mihaela, and van der, Schaar. 2022. HyperImpute: Generalized iterative imputation with automatic model selection. In *Proceedings of the 39th International Conference on Machine Learning*. Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.), Proceedings of Machine Learning Research, Vol. 162, PMLR, 9916–9937. Retrieved from https://proceedings.mlr.press/v162/jarrett22a.html

[25] Kaggle. 2016. House Sales in King County, USA. Kaggle. Retrieved from https://www.kaggle.com/datasets/harlfoxem/housesalesprediction

[26] Hyun Kang. 2013. The prevention and handling of the missing data. *Korean Journal of Anesthesiology* 64, 5 (May 2013), 402–406. DOI: https://doi.org/10.4097/kjae.2013.64.5.402

[27] Tero Karras, Miika Aittala, Samuli Laine, and Timo Aila. 2024. Elucidating the design space of diffusion-based generative models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22)*. Curran Associates Inc., Red Hook, NY, Article 1926, 13 pages.

[28] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*. I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Curran Associates, Inc., Vol. 30. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

[29] Jayoung Kim, Jinsung Jeon, Jaehoon Lee, Jihyeon Hyeong, and Noseong Park. 2021. OCT-GAN: Neural ODE-based conditional tabular GANs. In *Proceedings of the Web Conference 2021 (WWW '21)*. ACM, New York, NY, 1506–1515. DOI: https://doi.org/10.1145/3442381.3449999

[30] Jayoung Kim, Chaejeong Lee, and Noseong Park. 2023. STaSy: Score-based tabular data synthesis. In *Proceedings of the 11th International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=1mNssCWt_v

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

[32] Trent Kyono, Yao Zhang, Alexis Bellot, and Mihaela van der Schaar. 2021. MIRACLE: Causally-aware imputation via learning missing data mechanisms. In *Advances in Neural Information Processing Systems*. A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (Eds.), Curran Associates, Inc. Retrieved from https://openreview.net/forum?id=GzeqcAUFGl0

[33] Chaejeong Lee, Jayoung Kim, and Noseong Park. 2023. CoDi: Co-evolving contrastive diffusion models for mixed-type tabular synthesis. In *Proceedings of the 40th International Conference on Machine Learning*. Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.), Proceedings of Machine Learning Research, Vol. 202, PMLR, 18940–18956. Retrieved from https://proceedings.mlr.press/v202/lee23i.html

[34] J. Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory* 37, 1 (1991), 145–151. DOI: https://doi.org/10.1109/18.61115

[35] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D. Plumbley. 2023. AudioLDM: Text-to-audio generation with latent diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*. Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.), Proceedings of Machine Learning Research, Vol. 202, PMLR, 21450–21474. Retrieved from https://proceedings.mlr.press/v202/liu23f.html

[36] Diganta Misra. 2020. Mish: A self regularized non-monotonic activation function. In *Proceedings of the 31st British Machine Vision Conference 2020 (Virtual Event) (BMVC '20)*. BMVA Press. Retrieved from https://www.bmvc2020-conference.com/assets/papers/0928.pdf

[37] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. 2021. PPFL: Privacy-preserving federated learning with trusted execution environments. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (Virtual Event) (MobiSys '21)*. ACM, New York, NY, 94–108. DOI: https://doi.org/10.1145/3458864.3466628

[38] Boris Muzellec, Julie Josse, Claire Boyer, and Marco Cuturi. 2020. Missing data imputation using optimal transport. In *Proceedings of the 37th International Conference on Machine Learning*. Hal Daumé III and Aarti Singh (Eds.), Proceedings of Machine Learning Research, Vol. 119, PMLR, 7130–7140. Retrieved from https://proceedings.mlr.press/v119/muzellec20a.html

[39] Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy*. IEEE, 739–753. DOI: https://doi.org/10.1109/SP.2019.00065

[40] Alexander Quinn and Nichol Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning*. Marina Meila and Tong Zhang (Eds.), Proceedings of Machine Learning Research, Vol. 139, PMLR, 8162–8171. Retrieved from https://proceedings.mlr.press/v139/nichol21a.html

[41] Richard Nock and Mathieu Guillame-Bert. 2022. Generative trees: Adversarial and copycat. In *Proceedings of the 39th International Conference on Machine Learning*. Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.), Proceedings of Machine Learning Research, Vol. 162, PMLR, 16906–16951. Retrieved from https://proceedings.mlr.press/v162/nock22a.html

[42] R. Kelley Pace and Ronald Barry. 1997. Sparse spatial autoregressions. *Statistics & Probability Letters* 33, 3 (1997), 291–297.

[43] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, 85 (2011), 2825–2830. Retrieved from http://jmlr.org/papers/v12/pedregosa11a.html

[44] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems*. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Curran Associates, Inc., Vol.

31. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

[45] Oona Rainio, Jarmo Teuho, and Riku Klén. 2024. Evaluation metrics and statistical tests for machine learning. *Scientific Reports* 14, 1 (13 Mar 2024), 6086. DOI: https://doi.org/10.1038/s41598-024-56706-x

[46] Prakhar Rathi and Arpan Mishra. 2019. Insurance Company. Kaggle. Retrieved from https://www.kaggle.com/datasets/prakharrathi25/insurance-company-dataset

[47] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10684–10695.

[48] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[49] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising diffusion implicit models. In *Proceedings of the International Conference on Learning Representations*.

[50] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-based generative modeling through stochastic differential equations. In *Proceedings of the International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=PxTIG12RRHS

[51] Daniel J. Stekhoven and Maintainer Daniel J. Stekhoven. 2013. Package 'missForest'. R Package Version 1, 21.

[52] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. 2021. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. In *Advances in Neural Information Processing Systems*. M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34, Curran Associates, Inc., 24804–24816. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2021/file/cfe8504bda37b575c70ee1a8276f3486-Paper.pdf

[53] Amirsina Torfi, Edward A. Fox, and Chandan K. Reddy. 2022. Differentially private synthetic medical data generation using convolutional GANs. *Information Sciences* 586 (2022), 485–500. DOI: https://doi.org/10.1016/j.ins.2021.12.018

[54] Svetlana Ulianova. 2020. Cardiovascular Disease. Kaggle. Retrieved from https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset

[55] Arash Vahdat, Karsten Kreis, and Kautz Jan. 2021. Score-based generative modeling in latent space. In *Advances in Neural Information Processing Systems*. M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34, Curran Associates, Inc., 11287–11302. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2021/file/5dca4c6b9e244d24a30b4c45601d9720-Paper.pdf

[56] Stef Van Buuren and Karin Groothuis-Oudshoorn. 2011. MICE: Multivariate imputation by chained equations in R. *Journal of Statistical Software* 45 (2011), 1–67.

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Ł. Ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30, Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[58] Alexander Vergara, Shankar Vembu, Tuba Ayhan, Margaret A. Ryan, Margie L. Homer, and Ramón Huerta. 2012. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical* 166–167 (2012), 320–329. DOI: https://doi.org/10.1016/j.snb.2012.01.074

[59] Jie Wang, Rui Gao, and Yao Xie. 2021. Two-sample test using projected Wasserstein distance. In *Proceedings of the 2021 IEEE International Symposium on Information Theory (ISIT)*, 3320–3325. DOI: https://doi.org/10.1109/ISIT45174.2021.9518186

[60] Yufeng Wang, Dan Li, Xiang Li, and Min Yang. 2021. PC-GAIN: Pseudo-label conditional generative adversarial imputation networks for incomplete data. *Neural Networks: The Official Journal of the International Neural Network Society* 141 (2021), 395–403.

[61] Bingyang Wen, Yupeng Cao, Fan Yang, Koduvayur Subbalakshmi, and Rajarathnam Chandramouli. 2022. Causal-TGAN: Modeling tabular data using causally-aware GAN. In *Proceedings of the ICLR Workshop on Deep Generative Models for Highly Structured Data*.

[62] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional GAN. In *Advances in Neural Information Processing Systems*. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32, Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2019/file/254ed7d2de3b23ab10936522dd547b78-Paper.pdf

[63] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2018. GAIN: Missing data imputation using generative adversarial nets. In *Proceedings of the 35th International Conference on Machine Learning*. Jennifer Dy and Andreas Krause (Eds.), Proceedings of Machine Learning Research, Vol. 80, PMLR, 5689–5698.

[64] Bowen Zhang, Shuyang Gu, Bo Zhang, Jianmin Bao, Dong Chen, Fang Wen, Yong Wang, and Baining Guo. 2022. StyleSwin: Transformer-based GAN for high-resolution image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11304–11314.

[65] Hengrui Zhang, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. 2024. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *Proceedings of the 12th International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=4Ay23yeuz0

[66] Yishuo Zhang, Nayyar A. Zaidi, Jiahui Zhou, and Gang Li. 2021. GANBLR: A tabular data generation model. In *Proceedings of the 2021 IEEE International Conference on Data Mining (ICDM)*, 181–190. DOI: https://doi.org/10.1109/ICDM51629.2021.00103

[67] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y. Chen. 2021. CTAB-GAN: Effective table data synthesizing. In *Proceedings of the 13th Asian Conference on Machine Learning*. Vineeth N. Balasubramanian and Ivor Tsang (Eds.), Proceedings of Machine Learning Research, Vol. 157, PMLR, 97–112. Retrieved from https://proceedings.mlr.press/v157/zhao21a.html

[68] Shuhan Zheng and Nontawat Charoenphakdee. 2022. Diffusion models for missing value imputation in tabular data. In *Proceedings of the NeurIPS 2022 1st Table Representation Workshop*. Retrieved from https://openreview.net/forum?id=4q9kFrXC2Ae

Appendices

A Hyperparameter Tuning

Table A1. Discriminative Models: Hyperparameters Search Space

Model	Hyperparameter	Possible Values
XGBoost	max depth	$[1, 9]$
	learning rate	$[0.01, 1.0]$
	estimators	$[50, 500]$
	min child weight	$[1, 10]$
	gamma	$[10^{-8}, 1]$
	subsample	$[0.01, 1]$
	colsample bytree	$[0.01, 1]$
	reg alpha	$[10^{-8}, 1]$
	subsample	$[0.01, 1]$
	reg lambda	$[10^{-8}, 1]$
CatBoost	learning rate	LogUniform $[0.1, 1]$
	l2 leaf reg	LogUniform $[1, 100]$
	bagging temperature	LogUniform $[0.1, 20]$
	random strength	$[1.0, 2.0]$
	depth	$[1, 10]$
	min data in leaf	$[1, 300]$
LightGBM	max depth	$[3, 12]$
	learning rate	$[0.01, 1.0]$
	estimators	$[50, 500]$
	num leaves	$[20, 3{,}000]$
	min data in leaf	$[200, 10{,}000]$
	max bin	$[200, 300]$
	lambda l1	$[0, 100]$
	lambda l2	$[0, 100]$
	min gain to split	$[0, 15]$
	bagging fraction	$[0.2, 0.95]$
	feature fraction	$[0.2, 0.95]$
MLP	hidden layers	$[2, 4, 6, 8]$
	latent space size	$[64, 128, 256, 512]$
	batch size	$[64, 128, 256, 512, 1{,}024]$
	learning rate	LogUniform $[0.00001, 0.003]$
	epochs	500

Table A2. Generative Models: Hyperparameters Search Space

Model	Hyperparameter	Possible Values
TVAE	compress dims	[32, 64, 128, 256, 512]
	decompress dims	[32, 64, 128, 256, 512]
	embedding dim	[32, 64, 128, 256, 512]
	batch size	[64, 128, 256, 512, 1,024]
	learning rate	LogUniform[0.00001, 0.003]
	epochs	500
CTGAN	generator dim	[32, 64, 128, 256, 512]
	discriminator dim	[32, 64, 128, 256, 512]
	embedding dim	[32, 64, 128, 256, 512]
	batch size	[64, 128, 256, 512, 1,024]
	learning rate	LogUniform[0.00001, 0.003]
	epochs	500
CoDi	timesteps	[50]
	learning rate	$[2e-03, 2e-05]$
	dim(Emb(t))	[16, 32, 64, 128]
	$\{dim(h1),$	$\{\ \{16, 32, 64\},$
	$dim(h2),$	$\{32, 64, 128\},$
	$dim(h3)\}$	$\{128, 256, 512\}\ \}$
	λ_C	$[0.2, 0.3, \ldots, 0.8]$
	λ_D	$[0.2, 0.3, \ldots, 0.8]$
	epochs	500
Tabsyn	VAE-n Heads	[1, 2, 4]
	VAE-Factor	[16, 32, 64, 128]
	VAE-Layers	[1, 2, 3, 4]
	VAE-Learning Rate	LogUniform[0.00001, 0.003]
	VAE-Epochs	4,000
	Diffusion-MLP Denoising Dim	[512, 1,024, 2,048]
	Diffusion-Batch Size	[512, 1,024, 2,048, 4,096]
	Diffusion-Learning Rate	LogUniform[0.00001, 0.003]
	Diffusion-Epochs	10,000
TabDDPM	Timesteps	[100, 200, 300, 400, 600, 800, 1,000]
	latent space size	[64, 128, 256, 512, 1,024, 2,048, 4,096]
	mlp depth	[2, 4, 6, 8]
	batch size	[64, 128, 256, 512, 1,024]
	learning rate	LogUniform[0.00001, 0.003]
	epochs	500
MTabGen	timesteps	[100, 200, 300, 400, 600, 800, 1,000]
	latent space size	[64, 128, 256, 512]
	transformer layer num	[2, 3, 4]
	transformer heads	[2, 4, 8]
	transformer feedforward size	[256, 512]
	batch size	[64, 128, 256, 512, 1,024]
	learning rate	LogUniform[0.00001, 0.003]
	epochs	500

B Additional Experimental Results

In this appendix, we compare the training and sampling times, as well as the number of trainable parameters, of *MTabGen* against other tabular generative models, using the *Churn*, *California Housing*, and *Gas Concentrations* datasets as examples. The training and sampling times and the number of trainable parameters are highly influenced by both the model hyperparameters and the dataset characteristics (size and number of features).

Regarding model hyperparameters, each model has been optimized for these datasets as outlined in Section 6.1.1. This process involves using Bayesian optimization and Optuna to identify the best hyperparameters for each model, enhancing the ML efficiency described in Section 5.3.1. The specific hyperparameter search space for each model is provided in Table A2 in Appendix A, and Table B1 presents the training and sampling times, along with the number of trainable parameters, for the optimized models on the *Churn*, *California Housing*, and *Gas Concentrations* datasets.

Table B1. *Training/Sampling Time* and *Number of Trainable Parameters* Comparison for *Churn*, *California Housing*, and *Gas Concentrations* Datasets

	Churn			California Housing			Gas Concentrations		
	Training	Sampling	Parameters	Training	Sampling	Parameters	Training	Sampling	Parameters
TVAE	5 min 3 s	0.86 s	190,122	8 min 10 s	1.96 s	180,132	14 min 47 s	4.35 s	325,108
CTGAN	14 min 25 s	1.68 s	280,121	18 min 36 s	3.01 s	265,132	29 min 58 s	5.21 s	456,587
CoDi	2 h 26 min 5 s	6.64 s	615,212	2 h 54 min 10 s	10.54 s	567,987	3 h 58 min 15 s	16.38 s	953,578
Tabsyn	31 min 34 s	6.02 s	415,122	38 min 42 s	9.16 s	409,876	1 h 5 min 31 s	14.43 s	762,463
TabDDPM	14 min 36 s	12.75 s	221,363	18 min 12 s	18.21 s	209,815	29 min 36 s	28.48 s	420,589
MTabGen I	14 min 42 s	15.12 s	332,049	18 min 10 s	21.10 s	298,844	31 min 48 s	30.45 s	597,688
MTabGen II	14 min 53 s	15.30 s	332,049	18 min 42 s	22.95 s	298,844	32 min 24 s	35.12 s	597,688

In terms of dataset characteristics, *Churn*, *California Housing*, and *Gas Concentrations* differ in size (10,000 vs. 20,640 vs. 13,910 samples) and feature count (10 vs. 8 vs. 129), impacting absolute values such as training and sampling times, as well as the number of trainable parameters. Despite these differences, all the datasets yield consistent results.

From the results in Table 2, the baseline model that most closely matches the performance of *MTabGen I* and *MTabGen II* in terms of ML efficiency is *Tabsyn*. *Tabsyn* is a latent diffusion model where a VAE first projects tabular data into a dense, homogeneous continuous space, followed by a diffusion process. Compared to our model, *Tabsyn* has a greater number of trainable parameters and longer training times. However, in terms of sampling, *Tabsyn* outperforms our model. This advantage largely arises from the findings of [27], which suggest that the sampling process can be accelerated by reducing the number of backward diffusion steps through an effective choice of schedule and scale functions (see [27] for details). The schedule function determines the desired noise level at each diffusion timestep, while the scale function dictates how data scales over time. Their optimal selection aligns with that proposed by DDIM [50]. As highlighted by [27], this choice is independent of how the denoising model (e.g., the neural network) has been trained. Therefore, we plan to investigate the integration of our masking training mechanism with the sampling approach proposed by [27] in future work to improve the model's sampling speed.

A final comment addresses the impact of dynamic conditioning, specifically the differences between *MTabGen I* and *MTabGen II*. Table B1 demonstrates that the conditioning mechanism has a limited effect on sampling and training time, while the number of trainable parameters remains unchanged with or without conditioning.

This outcome arises because the conditioning mask only influences the input size of the encoder–decoder transformer, specifically affecting the sequence length of the transformer encoder

input (as illustrated in Figure 2 and discussed in Section 4.3). Notably, the number of trainable parameters in a transformer encoder does not depend on input length; therefore, *MTabGen I* and *MTabGen II* both retain the same parameter count. Although input sequence length can impact computational time, this effect is minimal in our case. Even with the largest dataset, consisting of 129 features, the encoder input remains a relatively short sequence of 129 tokens. Consequently, the impact on both sampling and training time is minimal.

Received 5 July 2024; revised 13 March 2025; accepted 16 May 2025