TAEGAN: Revisit GANs for Tabular Data Generation

Jiayu Li* LI.JIAYU@NUS.EDU.SG
National University of Singapore, Singapore; Betterdata AI, Singapore

Zilong Zhao* Z.ZHAO@NUS.EDU.SG
National University of Singapore, Singapore; Betterdata AI, Singapore

Kevin Yee KEVIN@BETTERDATA.AI
Betterdata AI, Singapore

Uzair Javaid UZAIR@BETTERDATA.AI
Betterdata AI, Singapore

Biplab Sikdar BSIKDAR@NUS.EDU.SG
National University of Singapore, Singapore

Editors: Hung-yi Lee and Tongliang Liu

Abstract

Synthetic tabular data generation has gained significant attention for its potential in data augmentation and privacy-preserving data sharing. While recent methods like diffusion and auto-regressive models (i.e., transformer) have advanced the field, generative adversarial networks (GANs) remain highly competitive due to their training efficiency and strong data generation capabilities. In this paper, we introduce Tabular Auto-Encoder Generative Adversarial Network (TAEGAN), a novel GAN-based framework that leverages a masked auto-encoder as the generator. TAEGAN is the first to incorporate self-supervised warmup training of generator into tabular GANs. It enhances GAN stability and exposes the generator to richer information beyond the discriminator's feedback. Additionally, we propose a novel sampling method tailored for imbalanced or skewed data and an improved loss function to better capture data distribution and correlations. We evaluate TAEGAN against seven state-of-the-art synthetic tabular data generation algorithms. Results from eight datasets show that TAEGAN outperforms all baselines on five datasets, achieving a 27% overall utility boost over the best-performing baseline while maintaining a model size less than 5% of the best-performing baseline model. Code is available at: https://github.com/BetterdataLabs/taegan.

1. Introduction

Synthetic data generation has gained tremendous attention due to its potential to provide an unlimited amount of data for data-hungry deep neural networks' training (de Wilde et al., 2024; Seib et al., 2020), and to unlock the use of a large amount of data with privacy and sensitivity concerns for sharing (EU, 2016). The prevalence of tabular data modality in real-world applications, from governmental system records to commercial transaction records, also poses a great demand for synthetic tabular data.

The research on synthetic tabular data generation using deep generative models initially focused on generative adversarial networks (GANs) (Goodfellow et al., 2014) and variational autoencoders (VAEs) (Kingma and Welling, 2014). In recent years, diffusion models (Ho

* Corresponding authors.

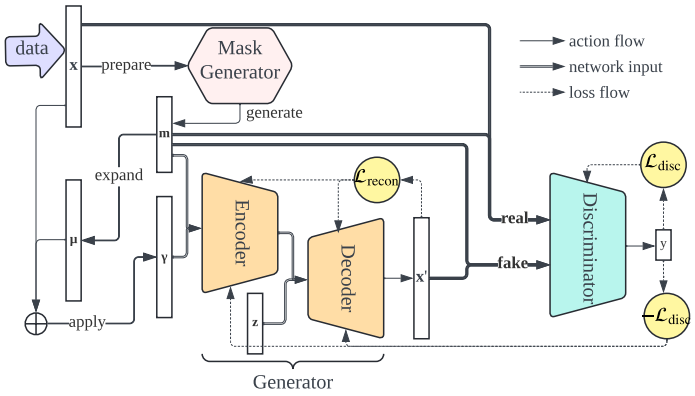

Figure 1: TAEGAN architecture. The generator is a masked auto-encoder. The encoder processes masked real data to produce an encoded vector, which is passed to the decoder along with noise. The mask and decoder output are concatenated for discrimination. A reconstruction loss is applied alongside the GAN losses.

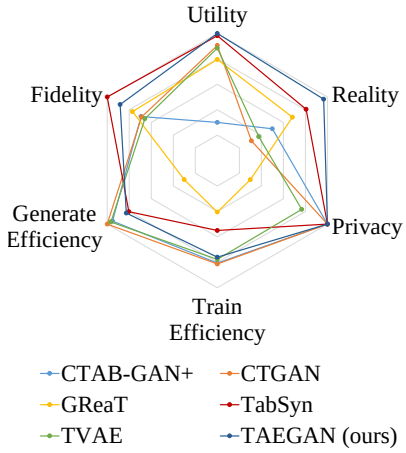

Figure 2: Comparison of TAEGAN with representative baseline models. TAEGAN demonstrates an overall advantage. over baseline models.

et al., 2020) and auto-regressive models (Brown et al., 2020) have gained significant traction due to their training stability and success in other modalities, such as images and text. However, GANs remain a competitive choice for tabular data generation, thanks to their unique advantages in training efficiency and generation speed (Xiao et al., 2022; Bond-Taylor et al., 2022), as well as their strong empirical performance—particularly given concerns that vision and language models may be over-skilled for tabular tasks (Gorishniy et al., 2021; Shi et al., 2021). In fact, a well-designed GAN can achieve comparable or even superior performance to diffusion and auto-regressive baselines while being far more efficient.

In this paper, we present such a design of tabular GAN: **T**abular **A**uto-**E**ncoder **G**enerative **A**dversarial **N**etwork (TAEGAN). Unlike the traditional design of tabular GANs where data are generated from noise or a sampled condition (Xu et al., 2019; Zhao et al., 2021), TAEGAN generates data from a masked version of sampled real data and gradually removes the mask by iterative decoding (Chang et al., 2022). Its generator is a modified auto-encoder with two additional inputs: i) a mask to the encoder for masked modeling, and ii) a noise vector to the decoder for the completeness of GAN. Besides the standard GAN loss, a reconstruction loss is also added. The overall architecture of TAEGAN is shown in Fig. 1. TAEGAN's generator is an independent auto-encoder, enabling self-supervised warmup training. The discriminator can be jointly trained in the warmup stage using the generator's output. This property not only improves the stability of GAN (Ham et al., 2020), which is a severe problem GANs have long suffered from, but also allows the generator to learn more about the real data by feedback beyond the discriminator's gradient update.

Besides the change in generative method and architecture, we also devise a novel data sampling approach during training to address the ubiquitous data imbalance and skewness problem. A novel loss component is also introduced to encourage the models to learn better data distributions and correlations.

Experiments comparing 7 state-of-the-art (SOTA) synthetic tabular data generation models over 8 datasets show TAEGAN's stable advantage in terms of synthetic data quality and efficiency, as shown in Fig. 2. Notably, although TAEGAN is less than 10% the

size of the best-performing baseline, it achieves a 27% performance improvement on the most widely used evaluation metric—machine learning efficacy (MLE). Furthermore, TAEGAN outperforms all baseline models in MLE. Additionally, thanks to the discriminator in the GAN framework, TAEGAN demonstrates a significant advantage in the reality of the synthetic data, as measured by the indistinguishability between real and synthetic data.

In summary, our major contributions include:

1. We introduce a novel tabular GAN framework, where the generator is a masked auto-encoder, enabling self-supervised warmup training. This approach enhances both training stability and the quality of the synthetic data.
2. We propose a *weight-matrix-based data sampling* method that addresses the common challenges of imbalance and skewness in tabular data during training.
3. We design a novel *information loss* that is designed specifically for tabular generative models to learn better data distribution and correlation.
4. We propose TAEGAN, which leverages the above three innovations, achieving SOTA performance in synthetic data quality while offering significantly improved efficiency compared to the top-performing baseline models.

2. Related Works

2.1. Generating Synthetic Tabular Data

When deep generative models were first introduced to the task of synthetic tabular data generation, GANs (Goodfellow et al., 2014) (e.g., CTGAN (Xu et al., 2019), CTAB-GAN (Zhao et al., 2021)) and VAEs (Kingma and Welling, 2014) (e.g., TVAE (Xu et al., 2019), GOGGLE (Liu et al., 2023)) were the predominant generative paradigms. In particular, CTGAN and TVAE (Xu et al., 2019) introduce tabular-specific feature engineering for these models and a data sampling method during training to address data imbalance. One-hot encoding is applied for categorical features, and variational Gaussian mixture (VGM) model decomposition is applied to continuous features to naturally capture multimodal distributions. Training data is sampled following the logarithmic frequency based on real-data-derived condition vectors. TAEGAN borrows or modifies these designs, with more details on necessary preliminaries seen in Section 3.

Recently, the increasingly popular generative models for image and text generation–diffusion models (Ho et al., 2020) and auto-regressive models (Radford et al., 2019; Brown et al., 2020), begin to dominate the field of synthetic tabular data generation too (e.g., TabDDPM (Kotelnikov et al., 2023), TabSyn (Zhang et al., 2024), and GReaT (Borisov et al., 2023)), due to the high quality of the synthetic data. However, these models, especially the better-performing ones, typically require a transformer (Vaswani et al., 2017) network that is usually used for image or text tasks. For example, TabSyn (Zhang et al., 2024) adapts a score-based diffusion model (Song et al., 2021; Karras et al., 2022) with a transformer architecture, which was initially designed for vision tasks, and GReaT (Borisov et al., 2023) adapts a Distil-GPT2 (Radford et al., 2019), which was initially designed for language tasks, with tabular-specific data encoding to generate synthetic tabular data.

However, while diffusion models and auto-regressive models generate synthetic data with impressive quality, the fact of using networks initially designed for images and text often implies a potential of over-fitting (Gorishniy et al., 2021; Shi et al., 2021). Moreover, the ef-

ficiency of diffusion models and auto-regressive models are typically worse than GANs (Xiao et al., 2022; Bond-Taylor et al., 2022). Therefore, there are still chances for a well-designed GAN specifically for tabular data to achieve comparable or better performance compared to these diffusion and auto-regressive models while being much more efficient.

2.2. Masked Modeling and Its Applications in Tabular Data

Masking is an important technique for self-supervised learning. Masked language modeling (MLM) has been a primary task for text pre-training (Devlin et al., 2019; Liu et al., 2019), and masked image modeling (MIM) has played a similar role in image pre-training (He et al., 2022; Bao et al., 2022). Data generation using masked modeling by gradually replacing masks with generated data has also gained great success in images (Chang et al., 2022).

In tabular data, masking is a natural method for imputation (Du et al., 2024). There are also generative models using masked modeling, such as TabMT (Gulati and Roysdon, 2023), which combines masking with a transformer. However, it suffers from long training and sampling time due to its iterative sampling nature on a large model.

3. Preliminaries and Notations

In this section, we introduce the notations and preliminaries from prior works that are necessary to understand TAEGAN. Most notations are a generalized version of prior works that helps the understanding of this paper.

3.1. Tabular Data Generation Notations

Problem Formulation. Given a real table $\mathcal{T}$ with M rows, N_d categorical and N_c continuous features ($N = N_d + N_c$ features in total), the objective is to generate synthetic $\mathcal{T}'$ of the same size that resembles the real one, i.e., $\mathcal{T}' \sim \mathcal{T}$.

Feature Engineering. TAEGAN inherits feature engineering from CTGAN (Xu et al., 2019). Categorical features are one-hot encoded. Continuous features are decomposed using VGM to capture multi-modal distributions. Each value is represented by concatenating the one-hot encoded mode index (e.g., peak on the probability density graph) with the numeric value within that mode (following a normal distribution) (see details in *mode-specific normalization* in (Xu et al., 2019)).

Tabular Notations. Thus, formally, each feature of the table can be represented as a vector $\mathbf{x}_n, n \in \{1, 2, \ldots, N\}$. $\mathbf{x}_n$ consists of a discrete *component* $\mathbf{d}_n$ (i.e., one-hot encoded category or mode for continuous feature) and a continuous *component* $\mathbf{c}_n$ (i.e., empty for categorical and the numeric value within the mode for continuous feature), namely, $\mathbf{x}_n = \mathbf{d}_n \oplus \mathbf{c}_n$, where $\oplus$ denotes vector concatenation. A categorical feature has only a discrete *component* and a continuous feature has a discrete and a continuous *component*. Then, let $|\cdot|$ be the number of dimensions of a vector, and let $D_n = |\mathbf{x}_n|, D_{dn} = |\mathbf{d}_n|, D_{cn} = |\mathbf{c}_n|$ ($D_n = D_{dn} + D_{cn}$). Thus, a row in $\mathcal{T}$ can be represented by the vector $\mathbf{x} = \bigoplus_{i=n}^{N} \mathbf{x}_n = \bigoplus_{n=1}^{N} \mathbf{d}_n \oplus \mathbf{c}_n$ with $C = N_d + 2N_c$ (non-empty) *components* and $D = \sum_{n=1}^{N} D_n$ dimensions. The "feature engineering" step in Fig. 3 shows an example of the representations.

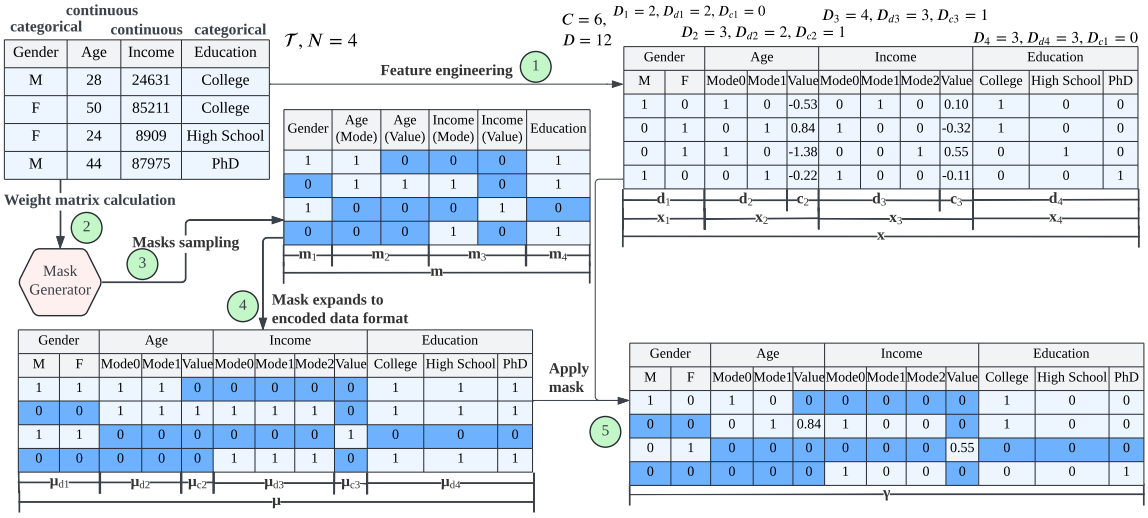

Figure 3: Example of TAEGAN data processing and mask process.

3.2. Conditional Vector

Prior work has found that tabular GANs perform better with a conditional generator (Xu et al., 2019). This requires a condition, we refer to as a *conditional vector*, passed together with noise to guide the generation of each row. The definition of *conditional vector*s for tabular GANs has evolved since its first introduction (Xu et al., 2019). In this section, we introduce a new representation compatible with different prior works and our proposed model TAEGAN.

We denote the *conditional vector* as $\boldsymbol{\gamma} = \bigoplus_{n=1}^{N} \boldsymbol{\gamma}_{dn} \oplus \boldsymbol{\gamma}_{cn}$, where $\boldsymbol{\gamma}_{dn}$ and $\boldsymbol{\gamma}_{cn}$ correspond to $\mathbf{d}_n$ and $\mathbf{c}_n$ respectively. In some prior works, not all components can be part of the condition. For example, CTGAN allows only components from categorical features. *Components* (α, can be dn or cn) disallowed in the condition have a corresponding $\boldsymbol{\gamma}_\alpha = \boldsymbol{\phi}, |\boldsymbol{\phi}| = 0$. Let the number of allowed components in the condition be $\widehat{C}$, and $\mathbf{x}$ with allowed components only be denoted $\widehat{\mathbf{x}}$. The *conditional vector* is essentially a masked $\widehat{\mathbf{x}}$. Each *component* is either fully masked or completely unmasked. Formally, let the *mask indicator* show the masking status for each *component* allowed in condition, represented as $\mathbf{m} = \bigoplus_{n=1}^{N} \mathbf{m}_{dn} \oplus \mathbf{m}_{cn} = (m_c)_{c=1}^{\widehat{C}} \in \{0,1\}^{\widehat{C}}$, where $|\mathbf{m}_\alpha| = 0$ if $\boldsymbol{\gamma}_\alpha = \boldsymbol{\phi}$ and $|\mathbf{m}_\alpha| = 0$ otherwise. In this paper, a mask value 1 means the value is maintained after masking. The *mask indicator* can be expanded by the dimensions of *components* to create the *data masks* as another binary vector $\boldsymbol{\mu} = \bigoplus_{i=1}^{N} \boldsymbol{\mu}_{di} \oplus \boldsymbol{\mu}_{ci}$, where $|\boldsymbol{\mu}_\alpha| = 0$ if $\boldsymbol{\gamma}_\alpha = \boldsymbol{\phi}$ otherwise $|\boldsymbol{\mu}_\alpha| = D_\alpha$, and $\boldsymbol{\mu}_{dn}, \boldsymbol{\mu}_{cn} \in \{\mathbf{0}, \mathbf{1}\}$ (vectors with all 0s or with all 1s). Then, the *conditional vector* is $\boldsymbol{\gamma} = \boldsymbol{\mu}^T \cdot \widehat{\mathbf{x}}$, which essentially means the masked data by the given *masked indicator*. Appendix A contains a summary of these notations to aid understanding.

Many prior works also have the constraint that exactly one *component* is maintained in the condition (Xu et al., 2019; Zhao et al., 2021), formally, $\|\mathbf{m}\|_1 = 1$ if $|\mathbf{m}| > 0$. In other words, $\mathbf{m} \in \{\mathbf{e}_i\}_{i=1}^{\widehat{C}}$ are one-hot vectors. Together with allowed *components* in condition, constraints on $\boldsymbol{\gamma}$ of prior tabular GANs and TAEGAN are compared in Table 1 and Fig. 4. In TAEGAN, all *components* are allowed in the condition, and hence $\mathbf{x} = \widehat{\mathbf{x}}$. In the rest of

Table 1: Comparison of the definition of *conditional vector*s in CT(GAN), CTAB(-GAN) and TAE(GAN), following the representation introduced in this paper.

	CT	CTAB	TAE	Remark
$\mathbf{m}_{cn}$	$=\boldsymbol{\phi}$	$=\boldsymbol{\phi}$	$\in\{0,1\}^1$	cont. comp. allowed in TAE
$\mathbf{m}_{dn}$, $D_{cn}>0$	$=\boldsymbol{\phi}$	$\in\{0,1\}^1$	$\in\{0,1\}^1$	disc. comp. from cont. feat. allowed in CTAB and TAE
$\|\mathbf{m}\|_1$, $\|\mathbf{m}\|>0$	$=1$	$=1$	$\in\{0,1\}^{\widehat{C}}$	no constraint for TAE

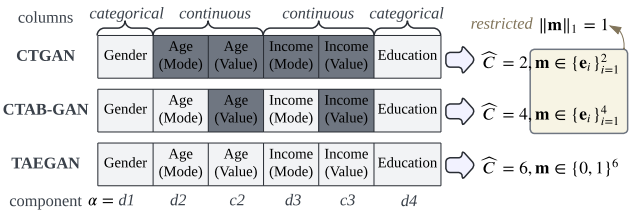

Figure 4: An example of *conditional vector*s of prior tabular GANs with TAEGAN.

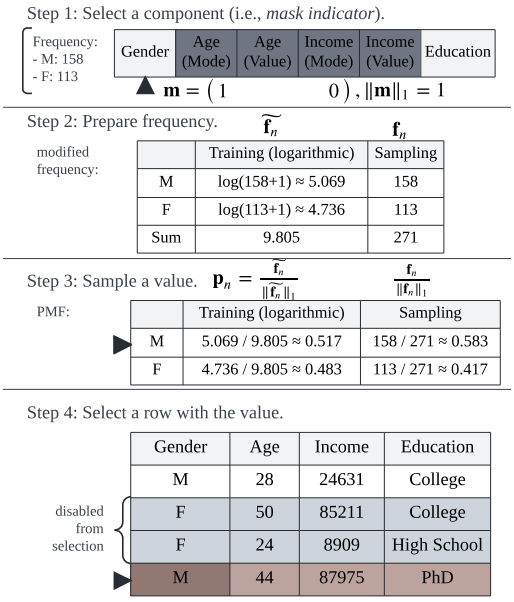

Figure 5: Data sampling of CTGAN explained with the example. Black arrows indicate the selected ones.

the paper, we use $\mathbf{x}$ for $\widehat{\mathbf{x}}$ without ambiguity. An example of TAEGAN *conditional vector* construction is visualized in Fig. 3.

3.3. Data Sampling with Logarithmic Frequency during Training

During training, CTGAN samples data by logarithmically transformed frequency of a value of a randomly selected feature. The *mask indicator* is first sampled, where only a categorical feature can remain unmasked. Then, with the *mask indicator* fixed, a value in the selected categorical feature is sampled by logarithmically transformed frequency. Formally, let $\mathbf{f}_n = (f_{ni})_{i=1}^{D_{dn}} \in \mathbb{N}^{D_{dn}}$ be the frequency vector of the n-th (categorical) feature in real data. Let the logarithmically transformed frequency vector smoothed by 1 be $\widetilde{\mathbf{f}_n} = \log(\mathbf{f}_n + 1)$. The probability mass function for sampling is thus a normalized version of $\widetilde{\mathbf{f}_n}$, i.e., $\mathbf{p}_n = \widetilde{\mathbf{f}_n}/\|\widetilde{\mathbf{f}_n}\|_1 = (p_{ni})_{i=1}^{D_{dn}}$. The process is visualized in Fig. 5

4. TAEGAN: Optimized GANs for Tabular Data

4.1. Overview and Architecture

Architecture. Recall the architecture of TAEGAN introduced in Fig. 1, the generator is a masked auto-encoder. The encoder takes in the *mask indicator* $\mathbf{m}$ and the masked data, i.e., *conditional vector* $\boldsymbol{\gamma} = \boldsymbol{\mu}^T \cdot \mathbf{x}$. The encoded data with the noise $\mathbf{z}$ is concatenated and passed to the decoder, which generates synthetic data $\mathbf{x}'$. Since most tabular data are a mixture of discrete and continuous values, we modify the noise $\mathbf{z}$ to be a vector whose first half is discrete with values 0's and 1's, and second half is continuous following standard normal distribution.

Algorithm 1 TAEGAN Training Algorithm

Data: Training table $\mathcal{T}$, warmup & main epochs E_p, E
Result: Generator parameters $\boldsymbol{\theta}_G$

```
Preprocess table 𝒯 by one-hot encoding and VGM decomposition
Calculate 𝒲 to prepare for sampling
Initialize network parameters θ_G, θ_D
for e in 1, 2, ..., E_p + E do
    repeat M times:
        x, m, μ, γ ← sampled data and masks ;                    /* Based on 𝒲 */
        Generate x' ← G(m, γ, z) based on noise z from latent space
        Update θ_G based on reconstruction loss, i.e., ℒ_recon(x', x, m)
        Discriminate the data y_r ← D(m, x), y_f ← D(m, x')
        Update θ_D based on discrimination, i.e., ℒ_disc(y_r, y_f)
        if e > E_p then
            /* Do the adversarial generation after warmup training */
            Update θ_G based on incorrect discrimination, i.e., −ℒ_disc(y_r, y_f)
```

Training. Since the generator is essentially an auto-encoder, a reconstruction loss can be calculated between $\mathbf{x}$ and $\mathbf{x}'$, allowing the generator to be trained without the discriminator, which we refer to as the warmup of GAN training. While the generator is trained in the warmup stage, the discriminator can be trained jointly but without direct interaction with the training of the generator in terms of gradients. Consequently, both the generator and discriminator can be warmed up independently, conceptually similar to a pre-training on the provided dataset. This allows a better initialization of both networks before the adversarial training of GAN starts, enabling improved stability during adversarial training (Ham et al., 2020). The overall training process is described in Algorithm 1, where details of some steps are elaborated in subsequent sections.

Generation. In TAEGAN, the masked auto-encoder design of the generator allows the mask $\mathbf{m}$ to be arbitrary as long as it falls in the range of $\{0, 1\}^C$, in contrast to CTGAN (Xu et al., 2019)'s mask with several constraints (recall Section 3.2). Consequently, a single row can be generated in multiple steps by iterative decoding, in contrast to CTGAN's one-time generation from the *conditional vector*. In the extreme case, one component is generated per step. The first component (step) can be directly taken from real data, which exposes only one component of one row, generally considered safe without privacy concerns as it contains a similar amount of information to the *conditional vector* of CTGAN. Although each time all dimensions are generated, we select only one additional from the input and proceed to the next iteration. The full process is visualized in Fig. 6, and details to be found in Appendix A.

4.2. Data Sampling during Training

Now that all constraints on $\mathbf{m}$ are relaxed (recall Section 3.2), the sampling of *mask indicator* and *conditional vector* are adjusted accordingly, to allow sampling based on conditions that are multivariate and potentially continuous.

Sampling the *Mask Indicator* $\mathbf{m}$. The *mask indicator* is sampled by first sampling a number of components (value of $\|\mathbf{m}\|_1$) from $\{1, 2, \ldots, C\}$, then randomly sampling this

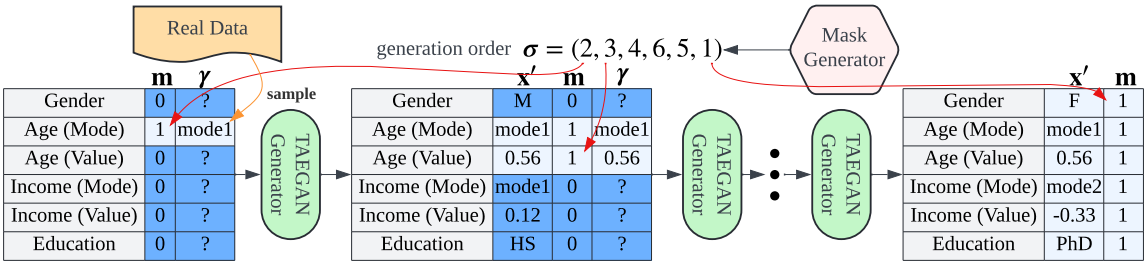

Figure 6: Generation process of TAEGAN. Components are generated one by one following a sampled order. The first component is sampled from real data.

Algorithm 2 *Weight Matrix* Computation

Data: Preprocessed training table $\mathcal{T}^*$
Result: *Weight Matrix* $\mathcal{W}$

1 **for** i **in** $1, 2, \ldots, C$ **do**
  `/* On` i`-th component (comp.) with` K `(binned) discrete values (val.) */`
2 $\mathbf{t} \in \{1, \ldots, K\}^M \leftarrow i$-th comp. of $\mathcal{T}^*$ by val. IDs
3 $\mathbf{f} \in (\mathbb{N}^+)^K \leftarrow$ frequency for each val.
4 $\widetilde{\mathbf{f}} \leftarrow \log(\mathbf{f} + 1)$; `/* Logarithmic transform */`
5 $\mathbf{p} \leftarrow \widetilde{\mathbf{f}} / \|\widetilde{\mathbf{f}}\|_1$; `/* Normalize to val. PMF */`
6 $\widetilde{\mathbf{p}} \leftarrow \mathbf{p}/\mathbf{f}$; `/* Normalize to row PMF */`
7 $\mathbf{w}_i \leftarrow \text{ONEHOT}(\mathbf{t}) \cdot \widetilde{\mathbf{p}}$; `/* Match PMF val. */`
8 $\mathcal{W} \leftarrow (\mathbf{w}_1 \quad \mathbf{w}_2 \quad \ldots \quad \mathbf{w}_C)$; `/* Combine all */`

Algorithm 3 TAEGAN Data Sampling during Training

Data: *Weight matrix* $\mathcal{W}$
Result: Sampled data and masks $\mathbf{x}, \mathbf{m}, \boldsymbol{\mu}, \boldsymbol{\gamma}$

1 $\mathbf{m} \leftarrow \mathbf{0}$
`/* Initialize an empty` ***mask indicator*** `*/`
2 $X \leftarrow$ sample $\|\mathbf{m}\|_1$ value following Equation 1
3 Update X non-repeated dimensions of $\mathbf{m}$ to 1
4 $\boldsymbol{\mu} \leftarrow$ expand $\mathbf{m}$ by dimensions per component
5 $\boldsymbol{\rho} \leftarrow \mathcal{W} \cdot (\mathbf{m}/\|\mathbf{m}\|_1)$
`/* Compute row-wise PMF under this` $\mathbf{m}$ `*/`
6 $\mathbf{x} \leftarrow$ sample a row based on $\boldsymbol{\rho}$
7 $\boldsymbol{\gamma} \leftarrow \boldsymbol{\mu}^T \cdot \mathbf{x}$

number of components. *Mask indicators* activating zero components (i.e., full mask) skipped during training because they are not used during generation (recall Algorithm 4). Also, note that it is generally harder to generate data with less information, so we sample smaller $\|\mathbf{m}\|_1$ with a higher probability mass than larger ones. In this paper, we use the normalized inverse of the values as the probability mass, namely, $\forall X \in \{1, 2, \ldots, C\}$,

$$\mathbb{P}(\|\mathbf{m}\|_1 = X) = \frac{1/X}{\sum_{i=1}^{C} 1/X} \tag{1}$$

Sampling the *Conditional Vector* $\boldsymbol{\gamma}$ and Data. Given a *mask indicator* without the constraints for CTGAN, the probability mass or density function of the corresponding *conditional vector* $\boldsymbol{\mu}$ cannot be computed by the feature-wise distribution as CTGAN does. Instead of sampling values based on the *mask indicator*, we sample real data rows and then apply the masks on them.

To calculate the probability for each row given the *mask indicator*, we compute the *weight matrix* $\mathcal{W} \in [0, 1]^{M \times C}$ as combination of column vectors $\mathbf{w}_i \in [0, 1]^M, i \in \{1, 2, \ldots, C\}$ with $\|\mathbf{w}_i\|_1 = 1$, which is the probability mass function over all rows in $\mathcal{T}$ when the i-th component is selected. Given the same *mask indicator*, rows with the same values after masking are sampled uniformly. Therefore, $w_{ij} = p_{nk}/f_{nk}$ if the i-th component is the discrete component of the n-th feature, and $\mathbf{d}_n = \mathbf{e}_k$ on the i-th row of $\mathcal{T}$. Continuous components' weights are calculated similarly by discretizing the values into bins so that frequencies and probability mass based on logarithmically transformed frequency can also be computed. The full process of *weight matrix* computation is described in Algorithm 2, and is used for Line 2 of Algorithm 1.

Finally, the dot product of $\mathcal{W}$ and normalized *mask indicator* $\mathbf{m}$, i.e., $\boldsymbol{\rho} = \mathcal{W} \cdot (\mathbf{m}/\|\mathbf{m}\|_1)$ indicate the probability mass function of each row to be sampled under this $\mathbf{m}$ by the multiplication law of probabilities. The full process is shown in Algorithm 3, and used for Line 6 of Algorithm 1. Appendix A provides an illustrative example to assist understanding.

Proposition 1 *The probability density/mass* $\mathbb{P}(\boldsymbol{\gamma}|\mathbf{m})$ *is identical from CTGAN (Xu et al., 2019) (recall Section 3.3) and from Algorithm 3, if* $\mathbf{m}$ *is allowed in both.*

Proposition 1 showcases the consistency of TAEGAN's design with prior works (e.g., CTGAN), but TAEGAN is a generalized version. The proof is seen in Appendix B.

4.3. Loss Functions

Reconstruction loss. Reconstruction loss can be computed on each component separately. We use cross-entropy for discrete and smooth L1 loss (Huber, 1964) for continuous components. We denote the loss for the i-th component as $\ell_i(\mathbf{x}', \mathbf{x}) \in \mathbb{R}$. The overall reconstruction loss is a weighted sum of losses on all components. We apply different weights on masked (known) and unmasked (unknown) components. Moreover, note that it is generally harder to reconstruct data from less known components because of the more diverse potential given limited known components, and easier from more known components as the reconstruction task eventually degenerates to classification or regression with increasing known components. Therefore, we adjust the weights on unknown components such that lower weights are applied when fewer components are known (smaller $\|\mathbf{m}\|_1$). We define a range of weights of unknown components as opposed to known components: $[\lambda_1, \lambda_2] \subseteq [0, 1]$, then the reconstruction loss on one sample is:

$$\mathcal{L}_{\text{recon}}(\mathbf{x}', \mathbf{x}, \mathbf{m}) = \sum_{i=1}^{C} \left(m_i + (1 - m_i) \left(\frac{\|\mathbf{m}\|_1 \cdot (\lambda_2 - \lambda_1)}{C} + \lambda_1 \right) \right) \cdot \ell_i(\mathbf{x}', \mathbf{x}) \tag{2}$$

The reconstruction loss is used to update the generator's gradients in both the warmup and main training stages. During warmup, this is the main loss function for the generator. During adversarial training, the reconstruction loss works in place of the auxiliary classification loss (Odena et al., 2017) that is also applied in some baseline tabular GANs (Zhao et al., 2021), as the reconstruction based on arbitrary masks is essentially a generalized version of the auxiliary classification task, and the generator network can be reused for classification to reduce the model size.

GAN loss. For better GAN training stability, we adopt WGAN-GP (Gulrajani et al., 2017) following CTAB-GAN (Zhao et al., 2021, 2024), and PacGAN (Lin et al., 2018) framework (i.e., the discriminator takes in "pacs" of, or a number of rows of, data) following CTGAN (Xu et al., 2019). However, for simplicity of higher-level understanding, we abstract them as $\mathcal{L}_{\text{disc}}(y_r, y_f)$ like a classical GAN (Goodfellow et al., 2014) in Algorithm 1.

Information loss. To capture a better data distribution, we adopt the additional information loss for the generator following CTAB-GAN (Zhao et al., 2021), which calculates the mean and standard deviations of the vector before the last layer of the discriminator of real and synthetic data respectively on every batch, and the loss is their differences. The information loss can be further enriched by direct calculations on $\mathbf{x}$ and $\mathbf{x}'$ utilizing properties

of the feature-engineered data, which consists of one-hot discrete components and normal-distributed continuous components (recall Section 3.1). Similarly to the original information loss, the losses are calculated by the differences of the aggregated values per batch between real and synthetic data. We design two additional loss components for marginal distribution and correlation respectively:

- **Mean loss for marginal distribution.** The aggregated value for *mean loss* is the mean for each dimension in the batch. In particular, the mean of one-hot vectors expresses exactly the probability mass function.
- **Interaction loss for correlation.** The *outer product* of two one-hot vectors of arbitrary dimensions (can be different dimensions) expresses a unique combination of the two input vectors, and the flattened vector is also one-hot. The *outer product* of a one-hot vector with a one-dimensional scalar also expresses a unique combination of the two input values if the result is not $\mathbf{0}$. Therefore, the correlation can be captured utilizing *outer product*s. The aggregated values for *interaction loss* are the mean and standard deviation of the flattened *outer product* $\mathbf{x} \otimes \mathbf{x}$ (supposing $\mathbf{x}$ is feature-engineered). In particular, dimensions in the *outer product* corresponding to impossible value combinations of two discrete components will always be 0, which can be fully captured by the aggregated values.

In summary, let $\mathbf{x}^{[k]}$ represent the k-th sample in a batch of size B, let $\Delta_{\text{mean}}(f(\mathbf{x}'), f(\mathbf{x}))$ where f is an aggregation function on $\mathbf{x}$ and $\mathbf{x}'$ denote the difference of mean aggregated values, i.e., $\|\textsc{Mean}_{k=1}^{B}(f(\mathbf{x}'^{[k]})) - \textsc{Mean}_{k=1}^{B}(f(\mathbf{x}^{[k]}))\|_1$, and let $\Delta_{\text{std}}(f(\mathbf{x}'), f(\mathbf{x}))$ be the difference of standard deviation similarly. Then, the modified information loss of a batch is

$$\begin{aligned}\mathcal{L}_{\text{info}}(\mathbf{x}'^{[1..B]}, \mathbf{x}^{[1..B]}) = \alpha_1 \cdot (\Delta_{\text{mean}}(\widetilde{D}(\mathbf{x}'), \widetilde{D}(\mathbf{x})) + \Delta_{\text{std}}(\widetilde{D}(\mathbf{x}'), \widetilde{D}(\mathbf{x})))\\ + \alpha_2 \cdot \Delta_{\text{mean}}(\mathbf{x}', \mathbf{x}) + \alpha_3 \cdot (\Delta_{\text{mean}}(\mathbf{x}' \otimes \mathbf{x}', \mathbf{x} \otimes \mathbf{x}) + \Delta_{\text{std}}(\mathbf{x}' \otimes \mathbf{x}', \mathbf{x} \otimes \mathbf{x}))\end{aligned} \tag{3}$$

where $\widetilde{D}$ represents the discriminator before the last layer, and $\alpha_1, \alpha_2, \alpha_3$ are the weights on the original information, mean, and interaction losses respectively.

Overall Objective. Let $p_{\mathbf{z}}$ be the probability distribution of the mixed-discrete-continuous noise $\mathbf{z}$, $p_{\mathbf{m}}$ is the probability distribution of $\mathbf{m}$ and $p_{\mathbf{x}|\mathbf{m}}$ is the probability distribution computed based on $\mathcal{W}$ given $\mathbf{m}$ as described in Section 4.2. Combining these losses, we have the objective of TAEGAN as

$$\begin{aligned}\min_G \max_D \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}, \mathbf{m}\sim p_{\mathbf{m}}} \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}|\mathbf{m}}}\\ (\mathcal{L}_{\text{recon}}(G(\mathbf{z}, \boldsymbol{\gamma}), \mathbf{x}, \mathbf{m}) - \mathcal{L}_{\text{disc}}\left(D(\mathbf{m}, \mathbf{x}), D(\mathbf{m}, G(\mathbf{z}, \boldsymbol{\gamma})\right) + \mathcal{L}_{\text{info}}(G(\mathbf{z}, \boldsymbol{\gamma}), \mathbf{x}))\end{aligned} \tag{4}$$

5. Experiments

5.1. Experiment Setup

Datasets. We conduct experiments on 8 datasets from OpenML (Vanschoren et al., 2013): `adult`, `bank-marketing`, `breast-w`, `credit-g`, `diabetes`, `iris`, `qsar-biodeg`, and `wdbc`. Details can be found in Appendix C.1.

Baselines. We compare the performance of TAEGAN with several representative tabular generative models as baselines, including non-neural-network models, GANs, VAEs, diffusion models, and auto-regressive models: ARF (Watson et al., 2023), CTAB-GAN+

(CTAB+) (Zhao et al., 2024), CTGAN (Xu et al., 2019), TabDDPM (TDDPM) (Kotelnikov et al., 2023), GReaT (Borisov et al., 2023), TabSyn (Zhang et al., 2024), and TVAE (Xu et al., 2019). Implementation details can be found in Appendix C.2.

TAEGAN Configuration. We use multi-layer perceptrons (MLP) as both the generator and discriminator. The networks are warmed up for 50 epochs capped at 500 steps, and then trained with adversarial losses for 300 epochs capped at 3000 steps, with a batch size of 3000 and learning rate of 2×10^{-4}. More details can be found in Appendix C.3.

Testbed. Experiments were conducted on a Ubuntu 22.04 server with an Intel i9-13900K CPU, 125 GB of RAM, and an NVIDIA RTX 4090 GPU.

Metrics. We evaluate tabular generative models using utility, reality, fidelity, privacy, and efficiency, which are standard metrics in tabular data generation (Xu et al., 2019; Zhao et al., 2024; Borisov et al., 2023; Zhang et al., 2024; Dat, 2023). All results reported are based on 3 runs of each experiment. Detailed implementation of the metrics can be found in Appendix C.4-C.8.

- **Utility** is usually tied to a downstream task (e.g., classification). The established machine learning efficacy (MLE) metric (Xu et al., 2019) has synthetic data generated from a real training set tested on a hold-out real test set. We use 3 downstream models: logistic regression (LG), random forest (RF) (Breiman, 2001), and XGBoost (XGB) (Chen and Guestrin, 2016). Better generative models yield higher weighted AUC ROC MLE results.
- **Reality** means the indistinguishability of real and synthetic data (Dat, 2023). The same 3 ML models as utility are used for this binary classification task. AUC ROC is reported, with a lower score (above 0.5) means a higher difficulty to differentiate real from synthetic, hence better utility.
- **Fidelity** shows the cosmetic discrepancy between the real and synthetic data in marginal distribution (MD, i.e., feature-wise) and correlation (CR, i.e., pair-wise) (Dat, 2023; Zhang et al., 2024). Higher similarity between real and synthetic data means better data fidelity.
- **Privacy** is crucial when the synthetic data is used for privacy-preserving data sharing. We use the distance to the closest record (DCR) (Zhao et al., 2021) to evaluate the privacy. The DCR from synthetic data and from hold-out real (test) data to the real training data are compared. Privacy is preserved if the latter DCRs are no smaller than the former, tested by Mann-Whitney U Test (Mann and Whitney, 1947). Privacy is at risk when p-values are smaller than 0.05.
- **Efficiency** is demonstrated by model sizes and computation time. Under comparable performance, smaller models and faster computation are preferred.

5.2. Comparison to Baselines

Utility: MLE. Table 2 summarizes the MLE performances of different generators. TAEGAN demonstrates a better overall performance than all baseline models, and wins over GAN baselines by a significant margin. In particular, TAEGAN outperforms the best-performing baseline's relative error compared to real by a factor of 27%. Comparing the performances on different downstream models, TAEGAN shows a clearer advantage on more complex ones (RF, XGB) than simpler ones (LG). This also implies the effect of a masked auto-encoder and the interaction loss in capturing complex data correlations.

Table 2: MLE performance summary. The first row is the overall average score. "RE" stands for the average relative error with the score produced by real data. The next 3 rows are the average scores on different downstream models. The best scores are in bold and underlined, and the second-best scores are in bold.

	real	ARF	CTAB+	CTGAN	TDDPM	GReaT	TabSyn	TVAE	TAEGAN
All (↑)	0.930	0.907	0.750	0.895	0.850	0.868	**0.913**	0.890	**0.917**
RE (↓)	-	2.632%	19.217%	3.873%	8.604%	6.897%	**1.951%**	4.511%	**1.422%**
LG (↑)	0.928	0.909	0.749	0.896	0.880	0.870	**0.915**	0.882	**0.914**
RF (↑)	0.930	0.903	0.755	0.892	0.866	0.866	**0.912**	0.892	**0.918**
XGB (↑)	0.932	0.908	0.745	0.896	0.803	0.869	**0.912**	0.895	**0.919**

Table 3: RSD performance summary. The first row is the average score of all experiments, and the other rows are the averages using different models. The best scores are in bold and underlined, and the second best scores are in bold.

	ARF	CTAB+	CTGAN	TDDPM	GReaT	TabSyn	TVAE	TAEGAN
All (↓)	0.771	0.834	0.902	0.818	0.768	**0.723**	0.878	**0.666**
LG (↓)	**0.610**	0.711	0.829	0.745	0.696	0.636	0.724	**0.588**
RF (↓)	0.843	0.890	0.936	0.850	0.804	**0.743**	0.948	**0.676**
XGB (↓)	0.860	0.900	0.941	0.857	0.803	**0.789**	0.963	**0.732**

Reality: RSD. Table 3 summarizes the RSD performances of different generators. TAEGAN demonstrates a clear advantage over all baseline models. GANs outperforming diffusion models and auto-regressive models on RSD is unsurprising due to the inherent presence of a discriminator, which is essentially an RSD task using a neural network. However, the fact that baseline GANs cannot outperform certain models in other generation paradigms shows that prior tabular GAN models have left significant room for further optimization.

Fidelity: MD and CR. Table 4 shows the average MD and DC scores. Some baseline models outperform TAEGAN, likely due to the logarithmic frequency during training in TAEGAN, which can slightly distort the distribution. All GAN frameworks experimented adopt a certain training with logarithmic frequency, and TAEGAN is clearly the best GAN model, demonstrating the effectiveness of the information loss construction.

Privacy: DCR. Table 5 shows the number of datasets where DCRs suggest a risk of privacy leakage. TAEGAN do not pose outstanding privacy risks.

Efficiency: Computation Time. Table 6 shows the average training and generation time over all 8 datasets of different models. The state-of-the-art diffusion and auto-regressive models generally take much longer to train and generate, while GANs are much more efficient. The non-neural network model is fast to train but very slow to generate. The results verify the efficiency of GANs.

Efficiency: Model Size. When it comes to model sizes, the better-performing diffusion models and auto-regressive models typically involve a transformer, with 20M or 80M of parameters usually. In comparison, TAEGAN involves only networks of simple MLP structure and the total model size including the discriminator is less than 0.5M. This also showcases the outstanding efficiency of TAEGAN.

The detailed experimental results of all metrics can be found in Appendix D.1.

Table 4: Average MD and DC scores over all 8 datasets. The best scores are in bold and underline, and the second best scores are in bold. The best score among GANs (*) are shown with superscript *.

	ARF	CTAB+*	CTGAN*	TDDPM	GReaT	TabSyn	TVAE	TAEGAN*
MD (↑)	**0.925**	0.898	0.852	0.751	0.881	**<u>0.931</u>**	0.860	0.906*
CR (↑)	**0.904**	0.825	0.857	0.783	0.867	**<u>0.919</u>**	0.839	0.892*

Table 5: Number of datasets (among 8 in total) where the synthetic data has a risk of privacy leakage.

ARF	CTAB+	CTGAN	TDDPM	GReaT	TabSyn	TVAE	TAEGAN
0	0	0	1	3	0	1	0

5.3. Loss Dynamics

Fig. 7 shows the trend of losses on two datasets. The reconstruction loss hits a plateau during warmup and continues to decrease when the main training stage starts, validating the usefulness of GAN framework besides a standalone auto-encoder. The difference of discriminator output on real and synthetic data is generally stable, verifying the stability of TAEGAN training.

5.4. Ablation Study

Table 7: Ablation study results.

	MLE				Fidelity	
	LG	RF	XGB	All	MD	CR
TAEGAN	0.867	0.861	0.866	0.865	0.952	0.933
w/o warmup	0.861	0.848	0.857	0.855	0.954	0.928
w/o log. freq.	0.868	0.859	0.859	0.862	0.957	0.938
$\lambda_1 = \lambda_2 = 1$	0.859	0.858	0.852	0.857	0.953	0.933
w/o discrete noise	0.864	0.859	0.864	0.862	0.952	0.929
w/o info. loss	0.867	0.861	0.861	0.863	0.952	0.933

In our ablation study, we assess the impact of key design choices in TAEGAN by systematically removing components. We highlight the result of the following 5 settings: i) Warmup training is eliminated (**w/o warmup**) by reallocating warmup epochs to main training; ii) Data sampling during training is adjusted to uniform sampling (**w/o log. freq.**) rather than logarithmic frequency-based sampling; iii) Mask ratio is fixed at equal values ($\lambda_1 = \lambda_2 = \mathbf{1}$); iv); Noise representation is altered by converting discrete noise to continuous values (**w/o discrete noise**); v) Information loss is removed (**w/o info. loss**). Ablation experiments are carried out on 3 datasets: `adult`, `credit`, and `diabetes`. TAEGAN with full setting performs the best on MLE, validating the effectiveness of all design components. The effect of warmup (vs. w/o warmup) and adjusted weights by mask ratio (vs. $\lambda_1 = \lambda_2 = 1$) are particularly obvious. Fidelity scores are better without logarithmically transformed sampling (w/o log. freq.), which is consistent with our supposition in Section 5.2 on the reason for the suboptimal performance of TAEGAN in fidelity.

6. Conclusion

In this paper, we introduce TAEGAN, a novel GAN framework for tabular data generation that uses a masked auto-encoder as the generator. We also propose a logarithmic frequency-based sampling method to address data imbalance and skewness, along with an improved loss function for better distribution and correlation in TAEGAN. When compared to exist-

Table 6: Average training and generation times (in seconds) across all 8 datasets. Generation means generating the same amount of data as the training set.

	ARF	CTAB+	CTGAN	TDDPM	GReaT	TabSyn	TVAE	TAEGAN
Training	51.933	187.517	178.590	67.330	2660.705	1023.474	226.858	254.823
Generation	10.147	0.302	0.208	15.776	47.157	0.952	0.270	0.798

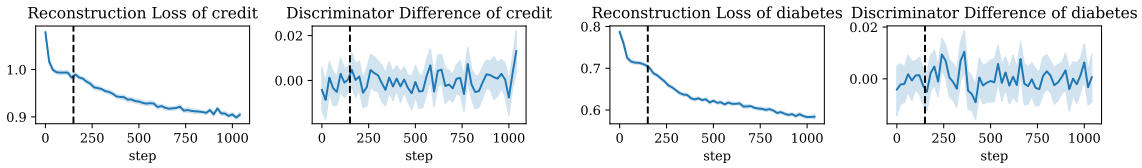

Figure 7: Dynamic of losses on `credit` and `diabetes`. "Reconstruction Loss" is in log10 scale for visibility. "Discriminator Difference" means $y_r - y_f$. The black vertical line is the divider between the warmup and main training stages.

ing baseline GANs and other tabular generative models across 8 datasets, synthetic data generated by TAEGAN achieves the highest utility in downstream tasks, closely resembles real data, while still effectively preserving privacy.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, et al. Optuna: A next-generation hyperparameter optimization framework. In SIGKDD, pages 2623–2631. ACM, 2019.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT pre-training of image transformers. In ICLR, 2022.

Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. Deep generative modelling: A comparative review of VAEs, GANs, normalizing flows, energy-based and autoregressive models. TPAMI, 44(11):7327–7347, 2022.

Vadim Borisov, Kathrin Sessler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. In ICLR, 2023.

Leo Breiman. Random forests. Machine learning, 45(1):5–32, 2001. ISSN 1573-0565.

Tom Brown, Benjamin Mann, Nick Ryder, et al. Language models are few-shot learners. In NeurIPS, volume 33, pages 1877–1901, 2020.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In CVPR, pages 11315–11325, June 2022.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In SIGKDD, pages 785–794. ACM, 2016.

Synthetic Data Metrics. DataCebo, Inc., December 2023. URL https://docs.sdv.dev/sdmetrics/. Version 0.13.0.

Philippe de Wilde, Payal Arora, Fernando Buarque, et al. Recommendations on the use of synthetic data to train AI models, 2024.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In NAACL-HLT, pages 4171–4186, Minneapolis, Minnesota, June 2019.

Tianyu Du, Luca Melis, and Ting Wang. ReMasker: Imputing tabular data with masked autoencoding. In ICLR, 2024.

EU. Regulation (EU) 2016/679 of the European parliament and of the council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/EC (General Data Protection Regulation). OJ, L 119, Apr 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. Generative adversarial nets. In NeurIPS, volume 27, 2014.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In NeurIPS, volume 34, 2021. ISBN 9781713845393.

Manbir Gulati and Paul Roysdon. TabMT: Generating tabular data with masked transformers. In NeurIPS, volume 36, 2023.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In NeurIPS, volume 30, 2017.

Hyungrok Ham, Tae Joon Jun, and Daeyoung Kim. Unbalanced GANs: Pre-training the generator of generative adversarial network using variational autoencoder, 2020. URL https://arxiv.org/abs/2002.02112.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In CVPR, pages 16000–16009, June 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In NeurIPS, volume 33, pages 6840–6851, 2020.

Peter J. Huber. Robust estimation of a location parameter. Ann. Math. Stat., 35(1):73–101, 1964.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, volume 37, pages 448–456. PMLR, 2015.

Tero Karras, Miika Aittala, Samuli Laine, and Timo Aila. Elucidating the design space of diffusion-based generative models. In NeurIPS, 2022. ISBN 9781713871088.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In ICLR, 2014.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. TabDDPM: Modelling tabular data with diffusion models. In ICML, volume 202, pages 17564–17579, 2023.

Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. PacGAN: The power of two samples in generative adversarial networks. In NeurIPS, volume 31, 2018.

Tennison Liu, Zhaozhi Qian, Jeroen Berrevoets, and Mihaela van der Schaar. GOGGLE: Generative modelling for tabular data by learning relational structure. In ICLR, 2023.

Yinhan Liu, Myle Ott, Naman Goyal, et al. RoBERTa: A robustly optimized BERT pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. Ann. Math. Stat., 18(1):50 – 60, 1947.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In ICML, volume 70, pages 2642–2651. PMLR, 2017.

Zhaozhi Qian, Rob Davis, and Mihaela van der Schaar. Synthcity: a benchmark framework for diverse use cases of tabular synthetic data. In NeurIPS, volume 36, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI, 2019.

Viktor Seib, Benjamin Lange, and Stefan Wirtz. Mixing real and synthetic data to enhance neural network training - a review of current approaches. 2020.

Xingjian Shi, Jonas Mueller, Nick Erickson, Mu Li, and Alexander J. Smola. Benchmarking multimodal AutoML for tabular data with text fields. In NeurIPS, volume 34, 2021. ISBN 9781713845393.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, et al. Score-based generative modeling through stochastic differential equations. In ICLR, 2021.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked science in machine learning. SIGKDD Explorations, 15(2):49–60, 2013.

Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. In NeurIPS, volume 30, page 6000–6010, 2017. ISBN 9781510860964.

David S. Watson, Kristin Blesch, Jan Kapar, and Marvin N. Wright. Adversarial random forests for density estimation and generative modeling. In 26th AISTATS, volume 206, pages 5357–5375, 2023.

Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion GANs. In ICLR, 2022.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. In NeurIPS, volume 32, 2019.

Hengrui Zhang, Jiani Zhang, Zhengyuan Shen, et al. Mixed-type tabular data synthesis with score-based diffusion in latent space. In ICLR, 2024.

Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y. Chen. CTAB-GAN: Effective table data synthesizing. In ACML, volume 157, pages 97–112, 2021.

Zilong Zhao, Aditya Kunar, Robert Birke, Hiek Van der Scheer, and Lydia Y. Chen. CTAB-GAN+: enhancing tabular data synthesis. Frontiers in Big Data, 6, 2024. ISSN 2624-909X.