

Towards Synthesizing High-Dimensional Tabular Data with Limited Samples

Zuqing Li, Junhao Gan, Jianzhong Qi*

School of Computing and Information Systems, The University of Melbourne
 {zuqingl@student, junhao.gan@, jianzhong.qi@}unimelb.edu.au

Abstract

Diffusion-based tabular data synthesis models have yielded promising results. However, when the data dimensionality increases, existing models tend to degenerate and may perform even worse than simpler, non-diffusion-based models. This is because limited training samples in high-dimensional space often hinder generative models from capturing the distribution accurately. To mitigate the insufficient learning signals and to stabilize training under such conditions, we propose CtrTab, a condition-controlled diffusion model that injects perturbed ground-truth samples as auxiliary inputs during training. This design introduces an implicit L_2 regularization on the model’s sensitivity to the control signal, improving robustness and stability in high-dimensional, low-data scenarios. Experimental results across multiple datasets show that CtrTab outperforms state-of-the-art models, with a performance gap in accuracy over 90% on average.

Code — <https://github.com/zuqingli0404/CtrTab>

Extended version — <https://arxiv.org/abs/2503.06444>

Introduction

Tabular data synthesis is an important problem with a wide range of applications. A common motivation is to facilitate privacy-preserving data sharing, i.e., to use synthetic data in scenarios where access to real data is restricted due to privacy concerns. In recent years, tabular data synthesis has also been used to help address data scarcity (Hsieh et al. 2025; Liu et al. 2024; Lu et al. 2023), augmenting training datasets to satisfy the need of modern machine learning models which are often data hungry. Meanwhile, the database community is using synthesized data for system performance benchmarking (Pang et al. 2024; Sanghi and Haritsa 2023; Yang et al. 2022). In this work, we focus on the non-privacy-sensitive settings, and we aim to synthesize data to enhance the performance of downstream tasks such as machine learning effectiveness (Zha et al. 2025).

Early studies on tabular data synthesis are primarily based on statistical models (Aggarwal and Yu 2004; Barak et al. 2007; Li et al. 2014; Park, Ghosh, and Shankar 2013; Zhang et al. 2014). With the rise of deep learning, models based

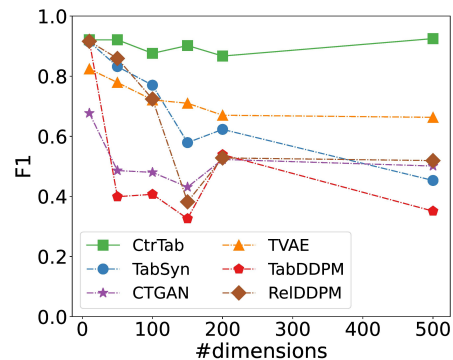


Figure 1: Challenges in tabular data synthesis over high-dimensional data. As the dimensionality increases, F1 scores of all existing models in machine learning tests decrease, while those of our model CtrTab remain stable.

on GANs and diffusion (Chen et al. 2019; Kim et al. 2021; Kotelnikov et al. 2023; Liu et al. 2024; Xu et al. 2019) are adopted. At the same time, to ensure the quality of synthesized data, studies introduce conditional generation (Liu et al. 2024; Xu et al. 2019; Zhao et al. 2023), incorporating additional information to guide the synthesis process.

Existing studies (Kim, Lee, and Park 2023; Kotelnikov et al. 2023; Lee, Kim, and Park 2023; Liu et al. 2024; Zhang et al. 2024) focus on datasets with a small number of columns (typically fewer than 50) and a large number of samples, where learning the underlying distribution is more tractable. A critical challenge underexplored is the difficulty posed by sparse, high-dimensional tabular data, i.e., tables with many (e.g., hundreds of) columns and only a few rows.

This situation, common in fields like biomedicine, suffers from the curse of dimensionality — as the number of features grows, data points become increasingly sparse in the feature space, and the distances between neighboring samples grow larger, making it difficult for models to accurately capture the underlying distribution. In addition, the limited number of samples poses the risk of overfitting, as models may memorize the training data instead of generalizing to unseen samples, further aggravating the challenge of reliable data synthesis in high-dimensional settings. Even the

*Corresponding author.

performance of the state-of-the-art (SOTA) diffusion model TabSyn (Zhang et al. 2024) for tabular data synthesis is still far from satisfactory under such settings. To verify this, we use data generation tools from `Scikit-learn` to generate tabular data, fixing the number of samples (rows) at 3,000 while varying the number of features (dimensions) from 10 to 500, with a class balance ratio of 0.5 for a binary classification task. For each dataset, we use 80% data to train recent tabular data synthesis models, and 20% for testing (following the machine learning tests detailed in the experimental section). Figure 1 plots the test results in F1 score. There is an overall decreasing trend in F1 for all models (except CtrTab which is ours), which is particularly obvious as the dimensionality reaches 500.

To address this issue, we propose **CtrTab**, a condition-controlled diffusion model designed for high-dimensional, low-data regimes. CtrTab introduces a control module alongside the denoising network. During training, each sample is perturbed with Laplace noise, serving as an auxiliary control signal. This control input is encoded and integrated into the decoder of the denoising network. Beyond improving controllability during generation, the control module provides an additional learning signal during training, enhancing the model’s robustness.

For better generalization under limited training samples, we employ a noise injection training strategy that systematically perturbs training samples. We theoretically show that this process is equivalent to introducing an implicit L_2 regularization on the model’s sensitivity to the control input, promoting smoother mappings and enhancing generalization beyond the sparse training data (Elman et al. 2020).

We conduct extensive experiments to show that while simple designs, e.g., perturbing data for augmentation or expanding model capacity, could offer some improvements, they fall short of addressing the underlying challenges. In contrast, CtrTab outperforms state-of-the-art models across various datasets, validating the effectiveness of our design.

Our contributions are summarized as follows: (1) We propose a diffusion-based tabular data synthesis model named CtrTab to address the challenge of sparse, high-dimensional data. Unlike existing diffusion-based models (Kotelnikov et al. 2023; Zhang et al. 2024) which struggle significantly under such settings, CtrTab introduces a *control module* and a *noise injection training strategy* that work together to improve model generalizability and robustness in complex tabular scenarios. (2) We provide a theoretical analysis showing that the proposed noise-based training is equivalent to L_2 regularization, where the noise scale flexibly controls the strength of regularization, enhancing model smoothness and stability. (3) We conduct experiments that extend tabular data synthesis to tables with up to 10,001 dimensions. The results show that machine learning models trained with data synthesized by CtrTab are much more accurate than those trained with data synthesized by SOTA models, with a performance gain of over 90% on average, confirming that CtrTab is more effective in learning the data distribution.

Related Work

Tabular Data Synthesis Recent advances in tabular data synthesis have shifted from statistical approaches (Aggarwal and Yu 2004; Barak et al. 2007; Li et al. 2014; Park, Ghosh, and Shankar 2013; Zhang et al. 2014) to deep generative models such as GANs (Chen et al. 2019; Kim et al. 2021; Park et al. 2018; Wen et al. 2022; Xu et al. 2019; Zhao et al. 2023), large language models (LLMs) (An et al. 2025; Wang et al. 2024) and diffusion models (Kim, Lee, and Park 2023; Kotelnikov et al. 2023; Lee, Kim, and Park 2023; Liu et al. 2024; Pang et al. 2024; Shi et al. 2024; Zhang et al. 2024; Si et al. 2025). While GAN-based models suffer from training instability (Arjovsky, Chintala, and Bottou 2017), diffusion- and LLM-based models (An et al. 2025; Wang et al. 2024) have shown improved sample quality and training robustness. Recent studies also consider missing-value imputation or minority-class data synthesis (D’souza, M, and Sarawagi 2025; Kim, Lee, and Park 2025; Schreyer et al. 2024).

Most existing diffusion models focus on low-dimensional, dense tabular data and do not generalize well to high-dimensional, low-sample scenarios. Our model represents a latest development of the diffusion-based models that addresses this gap.

Conditional Tabular Data Synthesis Conditional generative models have been proposed for tabular data synthesis with additional constraints. CTGAN (Xu et al. 2019) incorporates class labels in the generator to produce data conditioned on the target class. CTAB-GAN+ (Zhao et al. 2023) extends CTGAN to support both discrete and continuous data class labels. ReIDDP (Liu et al. 2024) uses a classifier-guided *conditional diffusion model*. It first trains an unconditional model to fit the input data distribution. Then, given a constraint, e.g., a target class label, it trains a classifier, the gradient of which is used to control sample generation. Our model does *not* concern class-conditioned generation. Instead, we introduce a control module to guide learning under sparse, high-dimensional conditions. There are also works utilizing LLMs (Fang et al. 2024; Wang et al. 2024). We aim for lighter-weight solutions and do not consider such solutions further.

Learning High-Dimensional Distribution with Sparse Data High-dimensional data poses significant challenges for machine learning. In *probably approximately correct* learning (Valiant 1984), the generalization error of a model depends on both the data dimensionality and the hypothesis class complexity. Typically, both the hypothesis space complexity and the required sample size grow exponentially as the dimensionality increases, making generalization more difficult. For generative models, increased dimensionality results in sparse data distributions (Bishop 2006), making it difficult for models to capture the underlying data structure.

Preliminaries

Problem Statement Consider a table \mathcal{T} of N rows, where each row x_{row} consists of D_{num} numerical features and D_{cat} categorical features that correspond to variables \mathbf{x}_{num} and

\mathbf{x}_{cat} , respectively. Categorical variables are encoded into numerical representations (e.g., one-hot encoding) before being used as model input. Let the dimensionality of all categorical variables after encoding be $E(D_{\text{cat}})$. The total model input data dimensionality is then $D = E(D_{\text{cat}}) + D_{\text{num}}$. Each row of \mathcal{T} , i.e., a data sample, is expressed as $\mathbf{x} = [\mathbf{x}_{\text{num}}, \mathbf{x}_{\text{cat}}]$, where $\mathbf{x}_{\text{num}} \in \mathbb{R}^{D_{\text{num}}}$ and $\mathbf{x}_{\text{cat}} \in \mathbb{R}^{E(D_{\text{cat}})}$.

Our aim is to train a generative model $p_{\theta}(\mathcal{T})$ such that the distribution of the generated data approximates that of the real data (i.e., rows) in \mathcal{T} . We focus on sparse, high-dimensional data, where $N \ll 2^D$. Here, 2^D represents the volume of the data space, which grows exponentially with D . We design our model based on diffusion models. Below, we briefly outline the core idea of diffusion models.

Denoising Diffusion Probabilistic Model The denoising diffusion probabilistic model (DDPM) (Ho, Jain, and Abbeel 2020) (see Figure 5 in the extended version) has two processes: a forward process and a reverse process, both Markov chains. In the **forward process**, noise is added to a data sample \mathbf{x}_0 from distribution $q(\mathbf{x}_0)$:

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}), \quad (1)$$

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}), \quad (2)$$

where t is the number of noise adding steps (timesteps), β_t controls the variance of the noise, $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$. In the **reverse process**, given a sample \mathbf{x}_t with pure random noise which is usually from a Gaussian distribution, a *denoising network* (i.e., the diffusion model) learns to iteratively denoise \mathbf{x}_t until it is restored to the initial sample \mathbf{x}_0 . The reverse process can also be expressed as a Gaussian distribution through derivation:

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t), \quad (3)$$

where $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0 + \frac{\sqrt{\bar{\alpha}_t} (1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t$ and $\tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t$.

The model learning process aims to maximize the variational lower bound:

$$\log(q(\mathbf{x})) \geq E_q \left[\underbrace{\log p_{\theta}(\mathbf{x}_0 | \mathbf{x}_1)}_{L_0} - \underbrace{D_{KL}(q(\mathbf{x}_T | \mathbf{x}_0) \| p(\mathbf{x}_T))}_{L_T} - \sum_{t>1} \underbrace{D_{KL}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{L_{t-1}} \right], \quad (4)$$

where $D_{KL}(\cdot)$ is the KL divergence, $q(\mathbf{x})$ the probability distribution of \mathbf{x}_0 , p_{θ} the parameterized model θ to approximate the probability distribution of the reverse process, and T the timestep at which noise is added, such that the original data distribution matches standard Gaussian distribution.

To maximize this lower bound means to minimize the KL divergence between $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ and $p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t)$. This objective further reduces to minimizing the sum of mean-squared errors between ϵ — the ground-truth error from \mathbf{x}_0 to \mathbf{x}_t — and ϵ_{θ} , the predicted noise by a denoising network:

$$L_{\text{simple}}(\theta) = \mathbb{E}_{\mathbf{x}_0, t, \epsilon} [\|\epsilon - \epsilon_{\theta}(\mathbf{x}_t, t)\|^2]. \quad (5)$$

Here, t and ϵ are generated randomly at training, while \mathbf{x}_t is from adding noise to \mathbf{x}_0 with Eq. (2).

Once trained, the denoising network ϵ_{θ} is used for data generation (called the sampling process). It takes \mathbf{x}_t and a timestep t as input, and its goal is to denoise \mathbf{x}_t iteratively back to \mathbf{x}_0 (i.e., a generated sample). Specifically, at each timestep t , it computes \mathbf{x}_{t-1} from \mathbf{x}_t as follows:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\theta}(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}, \quad (6)$$

where σ_t is the *noise scale* at timestep t , and \mathbf{z} follows the standard Gaussian distribution.

Methodology

As Figure 2 shows, CtrTab consists of two branches, a denoising network and a control module. The control module can be applied to different diffusion-based models – we use the SOTA model TabSyn (Zhang et al. 2024). With this module, CtrTab is trained on noisy data in addition to raw input to learn the input distribution more effectively, as will be shown by our theoretical analysis in the next section.

Denoising Network The denoising network takes as input a noisy sample \mathbf{x}_t (which comes from an encoded representation of ground-truth sample \mathbf{x}_0 during training and from standard Gaussian noise during sampling) and a timestep t . It outputs a predicted noise ϵ_{θ} .

We adopt TabSyn’s denoising network design and latent encoding strategy (detailed in the extended version as this is not our focus). Each raw sample x_{raw} from table \mathcal{T} is transformed into \mathbf{x}_0 using a model-agnostic embedding method. The hidden layers consist of three fully connected layers with SiLU activation functions. A final linear layer is applied to predict the noise. Note that our contributions focus on the control module and noise-based training (detailed next). The denoising network is modular and can be replaced with other networks (e.g., ReIDDPM (Liu et al. 2024) in our experiments).

Control Module We introduce a control module that receives a noisy version of the ground truth \mathbf{x}_0 as the conditioning input \mathbf{C}_f , enabling conditional generation to enhance model learning capability. This approach draws inspiration from classifier-free guidance (CFG) (Ho and Salimans 2021; Zhang, Rao, and Agrawala 2023), where conditions are injected into the diffusion model to guide data synthesis. Unlike CFG, we enable fine-grained control via an auxiliary network to inject conditions into the denoising process.

Noisy input. We use Laplace noise to form \mathbf{C}_f for its empirical effectiveness (detailed in experimental results):

$$\mathbf{C}_f = \mathbf{x}_0 + \text{Laplace}(b), \quad (7)$$

where $\text{Laplace}(b)$ represents the Laplace noise with mean 0 and variance $2b^2 - b$ is the *noise scale*. This noise-added input serves as a form of L_2 regularization during training, aiming to improve generalization with limited data.

The control module and the denoising network interact via *input fusion* and *control fusion*.

Input fusion. The control module and the denoising network share the same input \mathbf{x}_t and t , i.e., the sinusoidal timestep embedding (Kotelnikov et al. 2023; Liu et al. 2024; Zhang et al. 2024):

$$\mathbf{t}_{\text{emb}} = \text{Linear}(\text{SiLU}(\text{Linear}(\text{SinTimeEmb}(t))))), \quad (8)$$

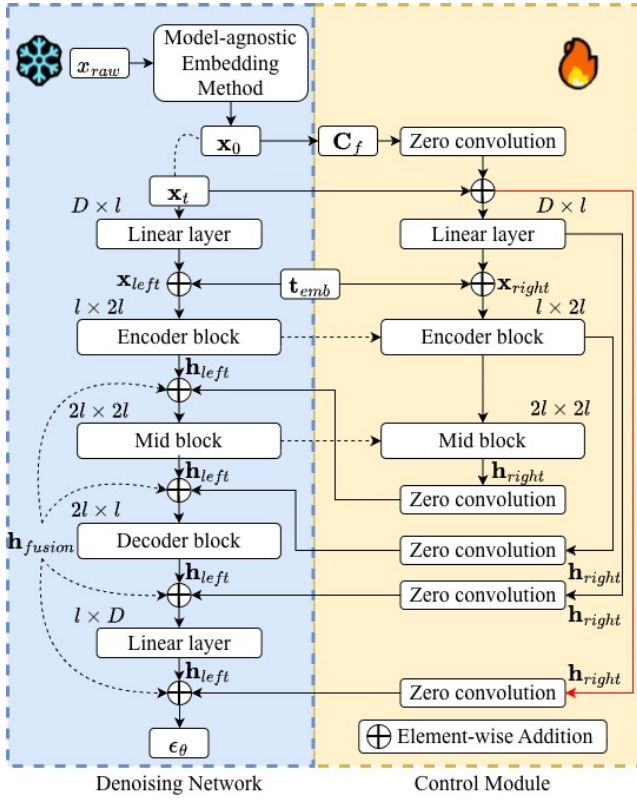


Figure 2: Overview of CtrTab. The left (blue) is a denoising network, which receives the noisy input \mathbf{x}_t and the timestep t , and predicts noise ϵ_θ . The right (yellow) is the control module, which encodes conditioning input \mathbf{C}_f and injects intermediate features via element-wise addition (\oplus). All fusion operations are followed by zero convolution to match dimensions. This modular design allows injecting conditioning signals without altering the diffusion backbone.

the timestep embedding is fused with \mathbf{x}_t for both modules:

$$\begin{aligned} \mathbf{x}_{\text{left}} &= \text{Linear}(\mathbf{x}_t) + \mathbf{t}_{\text{emb}}, \\ \mathbf{x}_{\text{right}} &= \text{Linear}(\text{zero_convolution}(\mathbf{C}_f) + \mathbf{x}_t) + \mathbf{t}_{\text{emb}}. \end{aligned} \quad (9)$$

Here, \mathbf{x}_{left} is the output of the top linear layer of the denoising network, and $\mathbf{x}_{\text{right}}$ is that of the top *zero convolution* and linear layers (see Figure 2). A zero convolution layer, implemented as a 1×1 convolution with zero-initialized weights and biases, is used to match dimensions and preserve alignment. The purpose is to learn effective encoded information of the condition before injecting it into the decoding layers of the denoising network to guide the output.

Control fusion. The embedding $\mathbf{x}_{\text{right}}$ then goes through an encoder block and a mid block, which are copied from the denoising network. The denoising network is pretrained and kept frozen when training the control module, to retain its strong data synthesis capability for denser, low-dimensional data. Let \mathbf{h}_{left} be the hidden outputs of the denoising network, and $\mathbf{h}_{\text{right}}$ be the output of the counterpart blocks in the control module. These outputs are fused as:

$$\mathbf{h}_{\text{fusion}} = \mathbf{h}_{\text{left}} + \text{zero_convolution}(\mathbf{h}_{\text{right}}). \quad (10)$$

Algorithm 1: Training

```

for each step do
  Sample  $\mathbf{x}_0 \sim q(\mathbf{x}_0), t$ , and  $\epsilon$ 
  Compute perturbed data  $\mathbf{x}_t$  and  $\mathbf{C}_f$ 
  Calculate loss  $\|\epsilon - \epsilon_\theta(\mathbf{x}_t, \mathbf{C}_f, t)\|^2$ 
  Update the control module via backpropagation
end for
return Trained CtrTab

```

Algorithm 2: Sampling

```

Sample  $\mathbf{x}_T, \mathbf{C}_f$ 
for  $t = T, \dots, 1$  do
  Predict  $\epsilon_\theta(\mathbf{x}_t, \mathbf{C}_f, t)$  using CtrTab output
  Obtain  $\mathbf{x}_{t-1}$  through the reverse equation of CtrTab
end for
return  $\mathbf{x}_0$ 

```

Note the red line in Figure 2, which connects the raw control module input directly to the output of the denoising network. This design preserves low-level condition information and complements the encoded high-level guidance, improving controllability and generation quality, especially in high-dimensional scenarios.

Training and Inference Our training objective is:

$$L_{DDPM}(\theta) = \mathbb{E}_{\mathbf{x}_0, t, \epsilon, \mathbf{C}_f} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, \mathbf{C}_f)\|^2. \quad (11)$$

We also support score-based generative models (Song et al. 2021) formulated as stochastic differential equations (SDEs), details can be found in the extended version.

As noted earlier, during training, gradients flow through the entire model, but only the control module’s parameters are updated while the denoising network remains frozen.

Algorithm 1 and 2 summarize the training and inference (i.e., sampling) of CtrTab, respectively.

Theoretical Results

We show that adding appropriate noise to the condition leads to a training objective analogous to one with the L_2 regularization, which is known to help prevent overfitting to the true data distribution (Ng 2004). For simplicity, in this section, we use \mathbf{C}_f to denote the noise-free condition, and $\mathbf{C}_f + \tilde{\epsilon}$ the noise-added condition. We present a summary of the theoretical results here and full proofs in the extended version.

Theorem 1. *Let the training objective of CtrTab be to minimize $\mathcal{L} = \mathbb{E}_{\mathbf{x}_0, t, \epsilon, \mathbf{C}_f} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, \mathbf{C}_f)\|^2$, and let the training objective with a noise added condition be to minimize $\tilde{\mathcal{L}} = \mathbb{E}_{\mathbf{x}_0, t, \epsilon, \mathbf{C}_f, \tilde{\epsilon}} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, \mathbf{C}_f + \tilde{\epsilon})\|^2$, where $\tilde{\epsilon}$ is a noise drawn from a distribution with mean 0 and variance η^2 . Then, $\tilde{\mathcal{L}} = \mathcal{L} + \eta^2 \mathcal{L}^R$ holds, where \mathcal{L}^R is a regularization term in Tikhonov form.*

Proof Sketch. Expansion of the training objectives. Following prior work (Bishop 1995; Webb 1994), we define $y(\mathbf{C}_f) := \epsilon_\theta(\mathbf{x}_t, t, \mathbf{C}_f)$ to simplify the notation. Then,

$\mathcal{L} = \mathbb{E}_{\mathbf{x}_0, t, \epsilon, \mathbf{C}_f} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, \mathbf{C}_f)\|^2$ can be expanded as:

$$\mathcal{L} = \int \int \int \int \{y(\mathbf{C}_f) - \epsilon\}^2 p(\epsilon | t, \mathbf{x}_0, \mathbf{C}_f) p(t, \mathbf{x}_0, \mathbf{C}_f) dt d\mathbf{x}_0 d\epsilon d\mathbf{C}_f. \quad (12)$$

Also, $\tilde{\mathcal{L}} = \mathbb{E}_{\mathbf{x}_0, t, \epsilon, \mathbf{C}_f, \tilde{\epsilon}} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t, \mathbf{C}_f + \tilde{\epsilon})\|^2$ becomes:

$$\tilde{\mathcal{L}} = \int \int \int \int \int \{y(\mathbf{C}_f + \tilde{\epsilon}) - \epsilon\}^2 p(\epsilon | t, \mathbf{x}_0, \mathbf{C}_f) p(t, \mathbf{x}_0, \mathbf{C}_f) p(\tilde{\epsilon}) dt d\mathbf{x}_0 d\epsilon d\mathbf{C}_f d\tilde{\epsilon}. \quad (13)$$

Some preliminaries. We expand $y(\mathbf{C}_f + \tilde{\epsilon})$ using a Second-order Taylor approximation, and assuming that y is three times continuously differentiable with bounded third derivatives, and that $\tilde{\epsilon}$ has a finite third moment, we have:

$$y(\mathbf{C}_f + \tilde{\epsilon}) \approx y(\mathbf{C}_f) + \sum_i \tilde{\epsilon}_i \left. \frac{\partial y}{\partial \mathbf{C}_{f,i}} \right|_{\tilde{\epsilon}=0} + \frac{1}{2} \sum_i \sum_j \tilde{\epsilon}_i \tilde{\epsilon}_j \left. \frac{\partial^2 y}{\partial \mathbf{C}_{f,i} \partial \mathbf{C}_{f,j}} \right|_{\tilde{\epsilon}=0} \quad (14)$$

This small-noise assumption is both theoretically and practically motivated. Practically, large-magnitude noise may overwhelm the original data features and hinder model training by masking meaningful patterns, as evidenced in Figure 3a, where the model’s performance drops when the noise scale increases. Theoretically, assuming small noise levels facilitates tractable analysis — a common practice in classical regularization literature (Bishop 1995).

By the fact that $\tilde{\epsilon}$ is chosen from a distribution with mean 0 and variance η^2 , we obtain:

$$\int \tilde{\epsilon}_i p(\tilde{\epsilon}) d\tilde{\epsilon} = 0, \int \tilde{\epsilon}_i \tilde{\epsilon}_j p(\tilde{\epsilon}) d\tilde{\epsilon} = \eta^2 \delta_{ij}, \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}. \quad (15)$$

The training objective with noise is a regularization. By substituting Eq. (14) into Eq. (13), we have:

$$\tilde{\mathcal{L}} = \mathcal{L} + \eta^2 \mathcal{L}^R, \text{ where} \quad (16)$$

$$\mathcal{L}^R = \int \int \int \int \sum_i \left\{ \left(\frac{\partial y}{\partial \mathbf{C}_{f,i}} \right)^2 + (y(\mathbf{C}_f) - \epsilon) \frac{\partial^2 y}{\partial \mathbf{C}_{f,i}^2} \right\} p(\epsilon | t, \mathbf{x}_0, \mathbf{C}_f) p(t, \mathbf{x}_0, \mathbf{C}_f) dt d\mathbf{x}_0 d\epsilon d\mathbf{C}_f. \quad (17)$$

The training objective with noise resembles L_2 regularization. We define two terms:

$$\langle \epsilon | t, \mathbf{x}_0, \mathbf{C}_f \rangle = \int \epsilon p(\epsilon | t, \mathbf{x}_0, \mathbf{C}_f) d\epsilon,$$

$$\langle \epsilon^2 | t, \mathbf{x}_0, \mathbf{C}_f \rangle = \int \epsilon^2 p(\epsilon | t, \mathbf{x}_0, \mathbf{C}_f) d\epsilon. \quad (18)$$

Then, Eq. (12) is expanded as follows:

$$\mathcal{L} = \int \int \int \int \int \{y(\mathbf{C}_f) - \langle \epsilon | t, \mathbf{x}_0, \mathbf{C}_f \rangle\}^2 p(\epsilon | t, \mathbf{x}_0, \mathbf{C}_f) p(t, \mathbf{x}_0, \mathbf{C}_f) dt d\mathbf{x}_0 d\epsilon d\mathbf{C}_f + \int \int \int \int \int \{ \langle \epsilon^2 | t, \mathbf{x}_0, \mathbf{C}_f \rangle - \langle \epsilon | t, \mathbf{x}_0, \mathbf{C}_f \rangle^2 \} p(\epsilon | t, \mathbf{x}_0, \mathbf{C}_f) p(t, \mathbf{x}_0, \mathbf{C}_f) dt d\mathbf{x}_0 d\epsilon d\mathbf{C}_f. \quad (19)$$

Observe that in Eq. (19), only the first integral term involves the parameters of a neural network (i.e., $y(\mathbf{C}_f) =$

$\epsilon_\theta(\mathbf{x}_t, t, \mathbf{C}_f)$, the denoising network to be trained), while the second term only depends on the ground-truth noise. Therefore, \mathcal{L} is minimized when $y(\mathbf{C}_f) = \langle \epsilon | t, \mathbf{x}_0, \mathbf{C}_f \rangle$.

Also, recall that $\tilde{\mathcal{L}} = \mathcal{L} + \eta^2 \mathcal{L}^R$ by Eq. (16). Consider the second term of \mathcal{L}^R in Eq. (17) and denote it as \mathcal{L}_2^R . Then, \mathcal{L}_2^R can be rewritten as follows:

$$\mathcal{L}_2^R = \int \int \int \int \sum_i \{y(\mathbf{C}_f) - \langle \epsilon | t, \mathbf{x}_0, \mathbf{C}_f \rangle\} \frac{\partial^2 y}{\partial \mathbf{C}_{f,i}^2} p(\epsilon | t, \mathbf{x}_0, \mathbf{C}_f) p(t, \mathbf{x}_0, \mathbf{C}_f) dt d\mathbf{x}_0 d\epsilon d\mathbf{C}_f. \quad (20)$$

Thus, when \mathcal{L} is minimized at $y(\mathbf{C}_f) = \langle \epsilon | t, \mathbf{x}_0, \mathbf{C}_f \rangle$, $\mathcal{L}_2^R = 0$. In this case, \mathcal{L}^R can be rewritten as:

$$\mathcal{L}^R = \int \int \int \sum_i \left(\frac{\partial y}{\partial \mathbf{C}_{f,i}} \right)^2 p(t, \mathbf{x}_0, \mathbf{C}_f) dt d\mathbf{x}_0 d\mathbf{C}_f, \quad (21)$$

which corresponds to the Tikhonov form. \square

Experiments

Datasets We use real-world datasets: **GE**, **CL**, **MA**, **ED**, **UN**, **UG**, and **EG**. These datasets, as summarized in Table 3 and detailed in the extended version, have been chosen for their large number of feature dimensions (up to 241) relative to the number of rows (a few thousands).

Competitors We compare CtrTab with six baseline models including state-of-the-art (SOTA) diffusion-based tabular data synthesis models: **SMOTE** (Chawla et al. 2002), **TVAE** (Xu et al. 2019), **CTGAN** (Xu et al. 2019), **TabD-DPM** (Kotelnikov et al. 2023), **ReIDDPM** (Liu et al. 2024), and **TabSyn** (Zhang et al. 2024) (SOTA). These models are detailed in the extended version, together with implementation details of these models and our CtrTab model.

Implementation Details All experiments were run three times on a virtual machine with a 12-core Intel(R) Xeon(R) Gold 6326 CPU (2.90 GHz), 32 GB of RAM, and an NVIDIA A100 GPU. We implement CtrTab and all competitor models using Python. For the competitors, we adopt the same hyperparameter settings as specified in the TabSyn paper (Zhang et al. 2024). We train CtrTab with the AdamW optimizer (Loshchilov and Hutter 2019) using a learning rate of 0.0018, weight decay of 0.00001 with 30,000 steps, and a noise scale of 0.005.

Results

Machine Learning Tests Following existing work (Kim, Lee, and Park 2023; Lee, Kim, and Park 2023; Zhang et al. 2024), we test the effectiveness of CtrTab mainly through machine learning tasks: (1) We train each model with the training set of each dataset. (2) Once trained, we use each model to synthesize a dataset of the same size of the respective training set. (3) We use the original training set and the synthesized set to separately train a downstream classifier or regression model (XGBoost and XGBoostRegressor, “**downstream model**” hereafter). (4) We evaluate these trained downstream models on the test set.

Following prior work (Zhang et al. 2024), we report classification results in the Area Under the ROC Curve (**AUC**)

Method	GE	CL	MA	ED	UN	Avg. Gap	UG	EG	Avg. Gap
	AUC \uparrow	AUC \uparrow	AUC \uparrow	AUC \uparrow	AUC \uparrow	% \downarrow	RMSE \downarrow	RMSE \downarrow	% \downarrow
Real	0.896	1	0.999	0.990	0.986	0%	0.036	0.115	0%
SMOTE	<u>0.865</u>	<u>0.999</u>	<u>0.996</u>	0.990	<u>0.976</u>	<u>0.98%</u>	<u>0.132</u>	<u>0.118</u>	<u>134.64%</u>
TVAE	-	-	-	0.825	0.819	16.80%	0.775	0.219	1071.61%
CTGAN	0.133	0.653	0.765	0.383	0.534	50.09%	0.754	0.266	1062.87%
TabDDPM	0.504	0.550	0.805	0.459	0.618	39.83%	2.216	0.346	3128.21%
RelDDPM	0.839	0.665	0.687	0.981	0.930	15.54%	0.463	0.122	596.10%
TabSyn	0.791	0.984	<u>0.996</u>	0.952	0.864	5.97%	0.267	0.166	343.01%
CtrTab (Ours)	0.890*	1*	0.999*	<u>0.988</u>	0.979*	0.32%	0.069*	0.114*	45.83%
	F1 \uparrow	F1 \uparrow	F1 \uparrow	F1 \uparrow	F1 \uparrow	% \downarrow	R2 \uparrow	R2 \uparrow	% \downarrow
Real	0.637	0.991	0.993	0.928	0.923	0%	0.998	0.646	0%
SMOTE	<u>0.598</u>	<u>0.968</u>	<u>0.991</u>	0.919	<u>0.889</u>	<u>2.66%</u>	<u>0.970</u>	<u>0.628</u>	<u>2.80%</u>
TVAE	-	-	-	0.673	0.604	31.02%	< 0	< 0	-
CTGAN	0.465	0.190	0.884	0.404	0.580	42.49%	0.031	< 0	96.89%
TabDDPM	0.092	0.018	0.872	0.546	0.620	53.98%	< 0	< 0	-
RelDDPM	0.554	0.334	0.871	0.905	0.841	20.60%	0.634	0.603	21.56%
TabSyn	0.450	0.845	0.983	0.849	0.711	15.32%	0.879	0.270	35.06%
CtrTab (Ours)	0.635*	0.983*	0.994*	<u>0.918</u>	0.898*	0.98%	0.991*	0.657*	0.35%

Table 1: Machine learning test results: The GE, CL, MA, ED, and UN datasets are used for classification, with results in AUC and F1. The UG and EG datasets are for regression, with results in RMSE and R2. Symbol ‘-’ indicates cases where the generative model collapsed, resulting in only a single class of generated samples, or negative R2 values such that the average gap becomes invalid. The best results are in boldface, while the second best are underlined. Symbol ‘*’ denotes values where CtrTab significantly outperforms *both* top baselines SMOTE and TabSyn ($p < 0.05$ in t-tests).

and **F1**, and regression results using the Root Mean Squared Error (**RMSE**) and R-Squared (**R2**).

Table 1 summarizes the results, where ‘‘Real’’ denotes the downstream model trained on the original data; each model (e.g., CtrTab) denotes the performance of the downstream model trained on data synthesized by that model. ‘‘Avg. Gap’’ denotes the (relative) performance gap between a model and ‘‘Real’’, averaged across the same type (classification or regression) of datasets. A smaller gap means a better model that better fits the original distribution.

Our model CtrTab reports the best results in almost all cases, except on ED, where SMOTE is marginally higher in AUC. The average gap in AUC, F1, RMSE, and R2 of CtrTab are 67.35%, 63.16%, 65.96%, and 87.50% lower than those of the best baseline model (SMOTE), respectively. Comparing with the SOTA model TabSyn, the gains are even higher, i.e., 94.64%, 93.60%, 86.64%, and 99.00%.

These results demonstrate the challenges for existing diffusion-based models to learn from sparse, high-dimensional data, and the effectiveness of CtrTab in addressing such challenges. SMOTE, while being an early model, shows high effectiveness, because it generates new samples through adding noise (directly) to existing samples which resembles our idea. TVAE and CTGAN are based on VAE and GAN, which also suffer from the data sparsity issue.

Ablation Study We further show the importance of the control module of CtrTab by comparing CtrTab with six alternative variants of TabSyn: (1) **Train** $\times 2$ doubles the number of training epochs for TabSyn. (2) **Data** $\times 2$ duplicates

each training sample with a Laplace noise of scale 0.01. (3) **Model** $\times 2$ duplicates each training sample as above and doubles the number of model parameters by expanding the hidden layers of TabSyn to match that of CtrTab (CtrTab is 1.8 \times the size of TabSyn). (4) **NoiseCond** applies the same noisy \mathbf{x}_0 signal (used in CtrTab’s control module) as an additional input to TabSyn, without structural changes. (5) **Dropout-Reg** applies Dropout with a rate of 0.1 to the hidden layers of TabSyn during training, serving as a standard regularization baseline. (6) **JointTrain** trains the control module and the diffusion model jointly in a single stage. This setting evaluates whether our staged training strategy contributes to performance improvements by better stabilizing the learning of the control signal. We also compare with **w/o-lastfusion**, i.e., CtrTab without the last h_{fusion} connection (the red line in Figure 2) to the denoising network, to verify the importance of this fusion connection.

We repeat the machine learning tests as above and report the results in F1 and R2 in Table 2. Results in AUC and RMSE share similar patterns and are in the extended version. CtrTab outperforms all these variants consistently, confirming the importance of the control module, our staged training strategy, and the last fusion connection.

Impact of Noise Type. We also replace Laplace noise in our control module with Gaussian and uniform noise. We find that while using Laplace noise typically leads to the best accuracy, the gain is often not too large. This confirms the robustness of CtrTab and the effectiveness of using Laplace noise. Detailed results are provided in the extended version.

Method	GE	CL	MA	ED	UN	UG	EG	Avg
Metric	F1↑	F1↑	F1↑	F1↑	F1↑	R2↑	R2↑	-
Train×2	0.443	0.831	0.988	0.852	0.698	0.860	0.390	0.746
Data×2	0.463	0.871	0.987	0.865	0.820	0.937	0.458	0.801
Model×2	0.522	0.872	0.988	0.892	0.816	0.884	0.569	0.822
NoiseCond	0.427	0.778	0.985	0.878	0.790	0.872	0.530	0.751
Dropout-Reg	0.395	0.794	0.989	0.835	0.697	0.812	0.429	0.707
JointTrain	0.525	0.970	0.977	0.857	0.732	0.766	0.569	0.772
w/o-lastfusion	0.619	0.982	0.992	0.909	0.883	0.805	0.655	0.835
CtrTab	0.635	0.983	0.994	0.918	0.898	0.991	0.657	0.868

Table 2: Ablation study results.

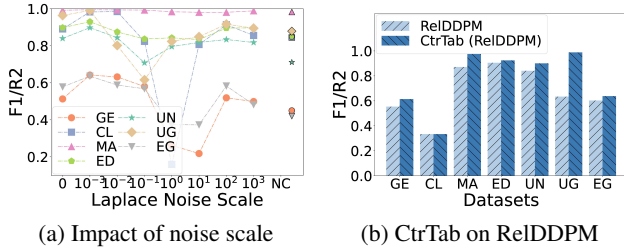


Figure 3: Impact of noise scale and the control module.

Parameter Study Figure 3a shows the downstream model accuracy (F1 and R2) when the Laplace noise scale in CtrTab is varied from 0 to 1000. As the noise scale increases, model accuracy decreases at start due to the impact of regularization by the noise scale (variance), which follows our theoretical results. When the noise grows further, the control module gradually fails, the regularization becomes ineffective, and the curves rise back until converging to the performance of the original diffusion model (i.e., CtrTab without the control module, “NC” in Figure 3a).

Figure 3a also shows results where the noise scale is 0, which means to train CtrTab without additive noise on the control signal C_f . We see that model performance drops comparing with using small Laplace noise as noted above. This supports our theoretical finding that noise injection implicitly imposes L_2 regularization, promoting smoother mappings and better generalization.

Control Module Applicability Our control module is not tied to TabSyn. We further integrate the control module with ReIDDPM (Liu et al. 2024), a representative DDPM-based model. ReIDDPM uses the standard diffusion forward and reverse processes (Eqs. 2 and 3) with a classifier-guided conditional generation. In our high-dimensional setting, the classifier is not required. We remove it and only adopt ReIDDPM’s denoising network to replace the denoising module in CtrTab. This effectively substitutes the SDE-based diffusion backbone with a DDPM-style architecture, while keeping the control module intact. The accuracy (F1 and R2) results are shown in Figure 3b, where CtrTab (with ReIDDPM) also outperforms ReIDDPM consistently. This verifies the applicability of our control module.

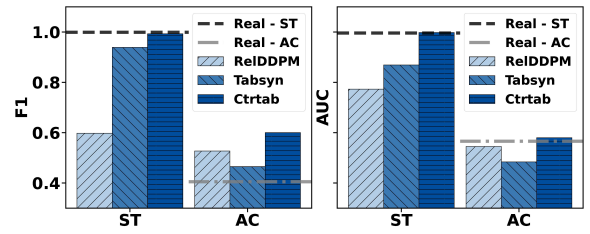


Figure 4: Case study on real-world extremely high-dimensional dataset. The dashed lines correspond to models trained on original data.

Case Study We explore an extremely high-dimensional setting uncommon in the literature, to assess model robustness. We test CtrTab on **ST** (UCI Machine Learning Repository 2019) and **AC** (Guyon et al. 2004) with 1,084 and 10,001 dimensions (4,998 and 100 rows, see the extended version), respectively, against the two diffusion baselines. Figure 4 shows downstream classification performance, confirming the robustness of CtrTab in this extreme setting.

On AC, the classifier trained on real data (“Real-AC”) performs poorly in F1, due to the presence of 3,000 noisy dimensions. In contrast, the synthetic samples, especially those generated by CtrTab, yield much higher F1 scores. This improvement is not only because the generative process captures more generalizable patterns and tends to suppress noisy or spurious correlations, but also because CtrTab is trained with implicit regularization, which encourages the generator to focus on informative structures in the data.

Real-AC in AUC is not as bad. AUC measures a model’s capability to rank positive and negative samples differently, while F1 is impacted by the classification threshold. As a result, AUC is less sensitive to noise and threshold selection.

Additional Results In the extended version, we further include results on model training and inference times, model performance with varying percentages of training data and on non-high-dimensional (i.e., conventional) datasets, distribution visualization results, results on distance to closest records and the impact of using different types of noise.

Conclusion

We proposed CtrTab, a model to enhance the fitting capability of diffusion generative models towards high-dimensional tabular data with limited samples. Through an explicit noise-conditioned control and training with a method similar to L_2 regularization, CtrTab synthesizes high-quality tabular data as evidenced by experiments over real datasets. The results show that machine learning models trained with data synthesized by CtrTab are much more accurate than those trained with data synthesized by existing models including the SOTA, with an accuracy gain of over 90% on average.

We extended tabular data synthesis to tables with up to 10,001 dimensions. Future work could scale to even higher dimensions. This work focuses on advancing the quality and controllability of tabular data synthesis in settings where privacy is not a constraint. Exploring how our methods interact with privacy constraints is left as future work.

Ethical Statement

All datasets used in this study are publicly available and do not contain any personally identifiable information. There are no ethical concerns associated with this work.

Acknowledgments

This work is in part supported by the Australian Research Council (ARC) via Discovery Projects DP230101534 and DP240101006. Jianzhong Qi is supported by ARC Future Fellowship FT240100170. Junhao Gan is in part supported by the ARC Discovery Project DP230102908.

References

- Aggarwal, C. C.; and Yu, P. S. 2004. A Condensation Approach to Privacy Preserving Data Mining. In *EDBT*, 183–199.
- An, S.; Woo, G.; Lim, J.; Kim, C.; Hong, S.; and Jeon, J.-J. 2025. Masked Language Modeling Becomes Conditional Density Estimation for Tabular Data Synthesis. In *AAAI*, 15356–15364.
- Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein Generative Adversarial Networks. In *ICML*, 214–223.
- Barak, B.; Chaudhuri, K.; Dwork, C.; Kale, S.; McSherry, F.; and Talwar, K. 2007. Privacy, Accuracy, and Consistency too. In *PODS*, 273–282.
- Bishop, C. M. 1995. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, 7(1): 108–116.
- Bishop, C. M. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag.
- Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; and Kegelmeyer, W. P. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16: 321–357.
- Chen, H.; Jajodia, S.; Liu, J.; Park, N.; Sokolov, V.; and Subrahmanian, V. S. 2019. FakeTables: Using GANs to Generate Functional Dependency Preserving Tables with Bounded Real Data. In *IJCAI*, 2074–2080.
- D’souza, A.; M, S.; and Sarawagi, S. 2025. Synthetic Tabular Data Generation for Imbalanced Classification: The Surprising Effectiveness of an Overlap Class. In *AAAI*, 16127–16134.
- Elman, M. R.; Minnier, J.; Chang, X.; and Choi, D. 2020. Noise Accumulation in High Dimensional Classification and Total Signal Index. *Journal of Machine Learning Research*, 21(36): 1–23.
- Fang, L.; Liu, A.; Zhang, H.; Zou, H. P.; Zhang, W.; and Yu, P. S. 2024. RES-RAG: Residual-aware RAG for Realistic Tabular Data Generation. In *NeurIPS 2024 Table Representation Learning Workshop*.
- Guyon, I.; Gunn, S.; Ben-Hur, A.; and Dror, G. 2004. Arcene. UCI Machine Learning Repository.
- Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising Diffusion Probabilistic Models. In *NeurIPS*, 6840–6851.
- Ho, J.; and Salimans, T. 2021. Classifier-Free Diffusion Guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.
- Hsieh, C.; Moreira, C.; Nobre, I. B.; Sousa, S. C.; Ouyang, C.; Brereton, M.; Jorge, J.; and Nascimento, J. C. 2025. DALL-M: Context-aware Clinical Data Augmentation with Large Language Models. *Computers in Biology and Medicine*, 190: 110022.
- Kim, J.; Jeon, J.; Lee, J.; Hyeong, J.; and Park, N. 2021. OCT-GAN: Neural ODE-based Conditional Tabular GANs. In *WWW*, 1506–1515.
- Kim, J.; Lee, C.; and Park, N. 2023. STaSy: Score-based Tabular Data Synthesis. In *ICLR*.
- Kim, J.; Lee, K.; and Park, T. 2025. To Predict or Not to Predict? Proportionally Masked Autoencoders for Tabular Data Imputation. In *AAAI*, 17886–17894.
- Kotelnikov, A.; Baranchuk, D.; Rubachev, I.; and Babenko, A. 2023. TabDDPM: Modelling Tabular Data with Diffusion Models. In *ICML*, 17564–17579.
- Lee, C.; Kim, J.; and Park, N. 2023. CoDi: Co-evolving Contrastive Diffusion Models for Mixed-type Tabular Synthesis. In *ICML*, 18940–18956.
- Li, H.; Xiong, L.; Zhang, L.; and Jiang, X. 2014. DPSynthesizer: Differentially Private Data Synthesizer for Privacy Preserving Data Sharing. *Proceedings of the VLDB Endowment*, 7(13): 1677–1680.
- Liu, T.; Fan, J.; Tang, N.; Li, G.; and Du, X. 2024. Controllable Tabular Data Synthesis Using Diffusion Models. *Proceedings of the ACM on Management of Data*, 2(1): 28:1–28:29.
- Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *ICLR*.
- Lu, Y.; Chen, L.; Zhang, Y.; Shen, M.; Wang, H.; Wang, X.; van Rechem, C.; Fu, T.; and Wei, W. 2023. Machine Learning for Synthetic Data Generation: A Review. arXiv:2302.04062.
- Ng, A. Y. 2004. Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance. In *ICML*, 78.
- Pang, W.; Shafieinejad, M.; Liu, L.; Hazlewood, S.; and He, X. 2024. ClavaDDPM: Multi-relational Data Synthesis with Cluster-guided Diffusion Models. In *NeurIPS*, 83521–83547.
- Park, N.; Mohammadi, M.; Gorde, K.; Jajodia, S.; Park, H.; and Kim, Y. 2018. Data Synthesis Based on Generative Adversarial Networks. *Proceedings of the VLDB Endowment*, 11(10): 1071–1083.
- Park, Y.; Ghosh, J.; and Shankar, M. 2013. Perturbed Gibbs Samplers for Generating Large-scale Privacy-safe Synthetic Health Data. In *IEEE International Conference on Healthcare Informatics*, 493–498.
- Sanghi, A.; and Haritsa, J. R. 2023. Synthetic Data Generation for Enterprise DBMS. In *ICDE*, 3585–3588.
- Schreyer, M.; Sattarov, T.; Sim, A.; and Wu, K. 2024. ImbFinDiff: Conditional Diffusion Models for Class Imbalance Synthesis of Financial Tabular Data. In *ACM International Conference on AI in Finance*, 617–625.

Shi, J.; Hua, H.; Xu, M.; Zhang, H.; Ermon, S.; and Leskovec, J. 2024. TabDiff: A Unified Diffusion Model for Multi-Modal Tabular Data Generation. In *NeurIPS 2024 Table Representation Learning Workshop*.

Si, J.; Ou, Z.; Qu, M.; Xiang, Z.; and Li, Y. 2025. TabRep: Training Tabular Diffusion Models with a Simple and Effective Continuous Representation. In *ICML Workshop on Foundation Models for Structured Data*.

Song, Y.; Sohl-Dickstein, J.; Kingma, D. P.; Kumar, A.; Ermon, S.; and Poole, B. 2021. Score-Based Generative Modeling through Stochastic Differential Equations. In *ICLR*.

UCI Machine Learning Repository. 2019. Malware Static and Dynamic Features VxHeaven and Virus Total.

Valiant, L. G. 1984. A Theory of the Learnable. *Communications of the ACM*, 27(11): 1134–1142.

Wang, Y.; Feng, D.; Dai, Y.; Chen, Z.; Huang, J.; Ananiadou, S.; Xie, Q.; and Wang, H. 2024. HARMONIC: Harnessing LLMs for Tabular Data Synthesis and Privacy Protection. In *NeurIPS*, 100196–100212.

Webb, A. 1994. Functional Approximation by Feed-forward Networks: A Least-squares Approach to Generalization. *IEEE Transactions on Neural Networks*, 5(3): 363–371.

Wen, B.; Cao, Y.; Yang, F.; Subbalakshmi, K.; and Chandramouli, R. 2022. Causal-TGAN: Modeling Tabular Data Using Causally-Aware GAN. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*.

Xu, L.; Skoularidou, M.; Cuesta-Infante, A.; and Veeramachaneni, K. 2019. Modeling Tabular Data Using Conditional GAN. In *NeurIPS*, 7335–7345.

Yang, J.; Wu, P.; Cong, G.; Zhang, T.; and He, X. 2022. SAM: Database Generation from Query Workloads with Supervised Autoregressive Models. In *SIGMOD*, 1542–1555.

Zha, D.; Bhat, Z. P.; Lai, K.-H.; Yang, F.; Jiang, Z.; Zhong, S.; and Hu, X. 2025. Data-centric Artificial Intelligence: A Survey. *ACM Computing Surveys*, 57(5): 129:1–129:42.

Zhang, H.; Zhang, J.; Srinivasan, B.; Shen, Z.; Qin, X.; Faloutsos, C.; Rangwala, H.; and Karypis, G. 2024. Mixed-Type Tabular Data Synthesis with Score-based Diffusion in Latent Space. In *ICLR*.

Zhang, J.; Cormode, G.; Procopiuc, C. M.; Srivastava, D.; and Xiao, X. 2014. PrivBayes: Private Data Release via Bayesian Networks. In *SIGMOD*, 1423–1434.

Zhang, L.; Rao, A.; and Agrawala, M. 2023. Adding Conditional Control to Text-to-Image Diffusion Models. In *ICCV*, 3836–3847.

Zhao, Z.; Kunar, A.; Birke, R.; Van der Scheer, H.; and Chen, L. Y. 2023. CTAB-GAN+: Enhancing Tabular Data Synthesis. *Frontiers in Big Data*, 6: 1296508.