Controllable Tabular Data Synthesis Using Diffusion Models

TONGYU LIU, Renmin University of China, China
JU FAN*, Renmin University of China, China
NAN TANG, HKUST (GZ), China
GUOLIANG LI, Tsinghua University, China
XIAOYONG DU, Renmin University of China, China

Controllable tabular data synthesis plays a crucial role in numerous applications by allowing users to generate synthetic data with specific conditions. These conditions can include synthesizing tuples with predefined attribute values or creating tuples that exhibit a particular correlation with an external table. However, existing approaches lack the flexibility to support new conditions and can be time-consuming when dealing with multiple conditions. To overcome these limitations, we propose a novel approach that leverages diffusion models to first learn an unconditional generative model. Subsequently, we introduce lightweight controllers to guide the unconditional generative model in generating synthetic data that satisfies different conditions. The primary research challenge lies in effectively supporting controllability using lightweight solutions while ensuring the realism of the synthetic data. To address this challenge, we design an unconditional diffusion model tailored specifically for tabular data. Additionally, we propose a new sampling method that enables correlation-aware controls throughout the data generation process. We conducted extensive experiments across various applications for controllable tabular data synthesis, which show that our approach outperforms the state-of-the-art methods.

CCS Concepts: • **Information systems → Data management systems**.

Additional Key Words and Phrases: Tabular Data Synthesis, Controllable Data Synthesis, Diffusion Model

ACM Reference Format:
Tongyu Liu, Ju Fan, Nan Tang, Guoliang Li, and Xiaoyong Du. 2024. Controllable Tabular Data Synthesis Using Diffusion Models. *Proc. ACM Manag. Data* 2, 1 (SIGMOD), Article 28 (February 2024), 29 pages. https://doi.org/10.1145/3639283

1 INTRODUCTION

Tabular data synthesis, which generates a synthetic table $\boldsymbol{R'}$ that closely resembles the distribution of an original table $\boldsymbol{R}$, has attracted significant interest recently [7]. Deep generative models, such as generative adversarial networks (GAN) [22] and auto-regressive autoencoders [21], have shown remarkable performance on synthetic data quality, and shed light on many applications, such as private data release [13, 17, 50], minority class oversampling [36], and data completion [27]. However, the existing approaches cannot naturally allow users to *control* the synthesis process to

*Ju Fan is the corresponding author.

Authors' addresses: Tongyu Liu, ltyzzz@ruc.edu.cn, Renmin University of China, Beijing, China; Ju Fan, fanj@ruc.edu.cn, Renmin University of China, Beijing, China; Nan Tang, nantang@hkust-gz.edu.cn, HKUST (GZ), Guangzhou, China; Guoliang Li, liguoliang@tsinghua.edu.cn, Tsinghua University, Beijing, China; Xiaoyong Du, duyong@ruc.edu.cn, Renmin University of China, Beijing, China.

ACM 2836-6573/2024/2-ART28
https://doi.org/10.1145/3639283

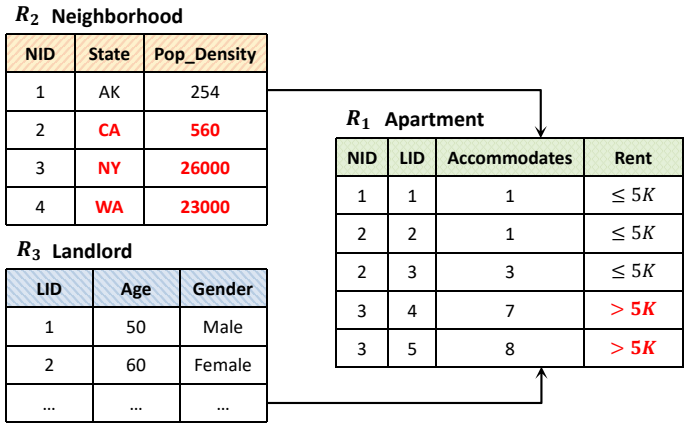


Fig. 1. An example of controllable tabular data synthesis for `Apartment`, which includes intra-table conditions (e.g., `Rent` $> 5K$) and inter-table conditions (e.g., tuples in `Neighborhood` and `Landlord`) to control the synthesis process.

generate synthetic data as desired, especially with respect to new conditions that have not been encountered during the training phase.

Controllable Tabular Data Synthesis. In many real-world applications, a key requirement for tabular data synthesis is the *controllability* of the synthesis process, which aims to generate synthetic table $\boldsymbol{R}'$ from original table $\boldsymbol{R}$ with some desired properties, called *conditions* in this paper. There are typically two types of conditions for common tabular data synthesis scenarios.

(1) Intra-table conditions focus on specific attributes or properties of original table $\boldsymbol{R}$, which impose constraints or requirements on attributes within synthetic table $\boldsymbol{R}'$. Consider an example table R_1 (`Apartment`) in Figure 1, where attribute `Rent` has two unique classes, `Rent` $\leq 5K$ and `Rent` $> 5K$. In this case, a user may want to impose an intra-table condition `Rent` $> 5K$ to generate a synthetic table $\boldsymbol{R}'_1$ that closely resembles the distribution of $\boldsymbol{R}_1$ with rent prices more than $5K$. A typical application of intra-table conditions is *oversampling* [11] that synthesizes tuples with the minority class to facilitate machine learning model training.

(2) Inter-table conditions consider using the attributes in an external table, say $\boldsymbol{R}_c$, as constraints to control the synthesis process, and the objective is to ensure the synthetic table $\boldsymbol{R}'$ to retain the correlation between the original table $\boldsymbol{R}$ and $\boldsymbol{R}_c$. As also shown in Figure 1, we consider table $\boldsymbol{R}_2$ (`Neighborhood`) that is joinable with $\boldsymbol{R}_1$ (`Apartment`) as $\boldsymbol{R}_c$. In this case, we can utilize tuples in $\boldsymbol{R}_2$ to control the synthesis process, such that the synthetic table $\boldsymbol{R}'_1$ should retain the correlation between $\boldsymbol{R}_1$ and the tuples from $\boldsymbol{R}_2$. For instance, neighborhoods with a higher population density tend to have apartments with higher rent prices. Thus, if we take the third tuple in $\boldsymbol{R}_2$ as a condition instance, the synthetic apartments are more likely to satisfy `Rent` $> 5K$. A typical application of inter-table conditions is data completion [27] that resembles the missing tuples of one table by utilizing complete tuples from other tables.

A well-adopted approach to controllable tabular data synthesis is to design a specific *conditional generative model* for each condition [27, 36, 47], and learn to control by providing the desired condition during *training*. Consider our example in Figure 1: suppose that we want to generate synthetic `Apartment` tuples conditioned on `Rent`. We need to first design a conditional generative model, such as conditional GAN [47], which takes values of `Rent` as an additional input, and then specify the conditions during training to control the synthesis process. However, there are two major drawbacks with the above approach.

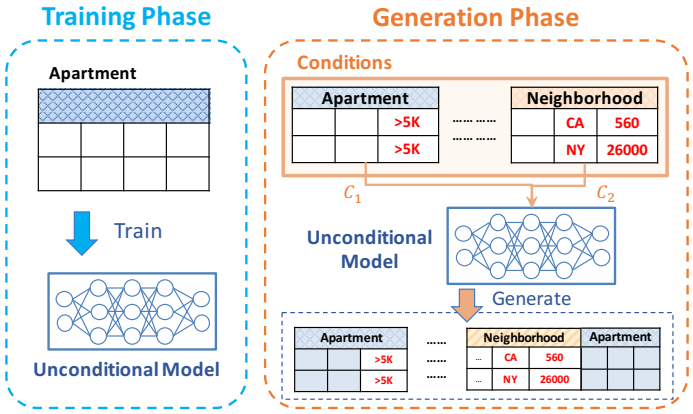


Fig. 2. An overview of our framework.

- *Flexibility:* the approach lacks flexibility as the condition is fixed for each conditional generative model, thus making it difficult to introduce new conditions without re-designing and re-training the model.
- *Efficiency:* training separate conditional generative models for each condition would be time-consuming, especially when dealing with multiple conditions.

Our Approach. To address the above problems, we propose a flexible and effective approach to support controllable tabular data synthesis, as illustrated in Figure 2. Instead of training a separate conditional generative model for each condition, the approach learns an unconditional generative model for the original table $\boldsymbol{R}$ to be synthesized during the training procedure. Then, during the generation procedure, it introduces a *controller* for each condition (such as C_1 and C_2 in Figure 2), which can effectively "guide" the unconditional generative model to generate the synthetic data satisfying the desired condition. Our approach offers several advantages over traditional methods. First, it allows for the introduction of new conditions during the generation process, not the training process, providing greater flexibility to different use cases and requirements. Second, since the unconditional generative model is already well-trained, we can maintain the high quality of synthetic data while efficiently generating under different conditions, thus mitigating the challenges of training conditional generative models from scratch.

Figure 2 illustrates how our approach supports controllable tabular data synthesis for our example table `Apartment` via two phases. In the training phase, a single unconditional generative model is learned from the original tuples in `Apartment`. In the generation phase, the unconditional generative model can be guided by our controllers to meet specified conditions, such as C_1 (generating synthetic data based on `Rent`) and C_2 (generating synthetic data based on external tuples from the `Neighborhood` table).

Challenges and Solutions. There are two key research challenges to support our proposed approach mentioned above.

(1) Unconditional Generative Model Learning. The first challenge is to train an unconditional generative model that not only resembles real tabular data but also can be easily adapted to specific conditions. Traditional generative models, such as GAN [17, 34, 64] and VAE [45, 54, 62] have been successfully used for learning unconditional tabular distribution, but they can not be controlled in the generation process. On the other hand, some very recent generative models, such as the Denoising diffusion probabilistic model (DDPM), have shown strong abilities to support controllable data generation [30, 40, 44, 70]. However, existing studies have shown that DDPM-based models cannot effectively resemble the distribution of tabular data [37].

(2) Correlation-aware Controller Construction. The second challenge is to design controllers to effectively guide the unconditional generative model to generate synthetic data for different conditions. The key difficulty is to ensure the controller captures correlations between the synthetic data and conditions. However, due to the high-dimensional and heterogeneous nature of tabular data, measuring such correlations is non-trivial.

Contributions. We propose a DDPM-based framework, namely RelDDPM, that effectively balances both the controllability and realism of the synthetic tabular data (Challenge 1). One key obstacle is that DDPM originally operates in the continuous space, and cannot be trivially applied in the tabular data with *mixed* data types, including categorical and numerical attributes. To address this, we introduce an analog bit encoding method to effectively represent tabular data in a continuous space, and develop diffusion model training algorithms for tabular data. To construct correlation-aware controllers in RelDDPM (Challenge 2), we propose a contrastive learning approach to embed the synthetic tuples and condition instances to a joint latent space, and then leverage the similarity of the embeddings between tuples and condition instances as a measurement of the correlation to control the generative model during the generation.

We summarize our contributions as follows:

(1) We introduce a novel framework for controllable tabular data synthesis, which can utilize controllers to guide a generative model to flexibly generate data that fulfills different conditions (Section 3).

(2) Building upon our framework, we design an unconditional DDPM specifically tailored for tabular data (Section 4). We apply analog bit encoding in tabular data synthesis using diffusion models and thus avoid additional diffusion processes for categorical attributes. We demonstrate that our designed diffusion model excels in preserving realism of the synthetic tabular data.

(3) We introduce a novel method based on contrastive learning to measure the correlation between condition instances and synthetic tuples, which can be used to provide correlation-aware controls over the generation process (Section 5).

(4) We have conducted extensive experiments, and the experimental results show that our method outperforms the state-of-the-art methods in various applications, and it is flexible and efficient to inject new conditions that have not been encountered during the training phase (Section 6). All the code and datasets in our experiments are public at the code repository[1].

2 PRELIMINARIES

2.1 Controllable Table Synthesis

The task of tabular data synthesis aims to train a generative model learned from a relational table $\boldsymbol{R}$ and use the model to generate a synthetic table $\boldsymbol{R}'$. More formally, table $\boldsymbol{R}$ contains m attributes $\{A_1, A_2, \dots A_m\}$ and n tuples $\{\boldsymbol{t}_1, \boldsymbol{t}_2, \dots, \boldsymbol{t}_n\}$. Each attribute is considered as a random variable, and these random variables (i.e., all attributes) follow an unknown real joint distribution $p_{A_1 \dots A_m}(v_1, v_2, \dots v_m)$, where v_j is a value designated for attribute A_j. Each tuple of $\boldsymbol{R}$, denoted by $\boldsymbol{t} = \{v_1, v_2, \dots, v_m\}$, can be regarded as one observation from the joint distribution. For simplicity, we denote the joint distribution as $p(\boldsymbol{t})$ if the context is clear.

Conditions. We consider a **condition** $\boldsymbol{C}$ associated with table $\boldsymbol{R}$, which is a set of k attributes $\{A_{c1}, A_{c2}, \dots, A_{ck}\}$ correlated with the attributes in $\boldsymbol{R}$. Each of these condition attributes is considered as a random variable, and the random variables in $\boldsymbol{R}$ follow an unknown conditional distribution $p_{\boldsymbol{R},\boldsymbol{C}}(\boldsymbol{t}|v_{c1}, v_{c2}, \dots, v_{ck})$ with variables from $\boldsymbol{C}$ as conditions, where v_{ci} denotes value

[1] https://github.com/ruc-datalab/RelDDPM

designated for attribute A_{ci}. A tuple $\boldsymbol{c} = \{v_{c1}, v_{c2}, \ldots, v_{ck}\}$ can be regarded as an instance of the condition $\boldsymbol{C}$. For simplicity, we denote the conditional distribution as $p(\boldsymbol{t}|\boldsymbol{c})$. Depending on the different sources of attributes in $\boldsymbol{C}$, there are typically two forms of conditions:

(1) **Intra-table conditions:** $\boldsymbol{C}$ is a set of attributes within table $\boldsymbol{R}$, which ensures that the generated tuples satisfy the specified criteria for the specified attributes in $\boldsymbol{R}$.
(2) **Inter-table conditions:** The attributes in $\boldsymbol{C}$ come from an external table that has a primary key-foreign key reference relationship with table $\boldsymbol{R}$, which aims to ensure that the generated data maintains the desired correlations across these two related tables.

Controllable Table Synthesis. Given a relational table $\boldsymbol{R}$ and a condition $\boldsymbol{C}$, suppose there are k condition instances $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k$, *controllable table synthesis* aims to synthesize a set of tuples $\boldsymbol{R}' = \{\boldsymbol{t}'_1, \boldsymbol{t}'_2, \ldots, \boldsymbol{t}'_k\}$ relative to the given condition instances. Each synthetic tuple $\boldsymbol{t}'_i \in \boldsymbol{R}'$ should satisfy two desired properties:

(1) *Resembling real data:* The synthetic tuple should accurately capture the individual attribute distributions and preserve the inherent patterns and associations among the attributes in the original table $\boldsymbol{R}$. Formally, we need to capture the unconditional distribution $p(\boldsymbol{t})$ for this property.
(2) *Meeting specified condition:* The synthetic tuple should satisfy the corresponding condition instance $\boldsymbol{c}_i$, i.e., ensuring that the synthetic tuple aligns with the desired correlation criteria. To this end, we have to capture the conditional distribution $p(\boldsymbol{t}|\boldsymbol{c})$ for this property.

Example 1 (Controllable Table Synthesis). [*Intra-table condition*] *Let us consider the Apartment table in Figure 1 as the table* $\boldsymbol{R}$. *An intra-table condition* $\boldsymbol{C}$ *could be* {`Rent`}, *which can be used to generate the tuples with specific rent prices. As depicted in Figure 2, we can provide a condition instance* $\boldsymbol{c}$ $(> 5K)$ *to specify that the* `rent` *values in the generated tuple should be greater than* $5K$. *According to the correlation between the rent price and accommodation, it is likely that the* `Accommodation` *of the synthetic tuple would have a large number, e.g.,* ≥ 7.

[Inter-table condition] Next, let us consider using the attributes in another table, `Neighborhood` *in Figure 1, as an inter-table condition* $\boldsymbol{C}$ *to control the generated tuples in* `Apartment` *located in a specific neighborhood. As shown in Figure 2, we can select the tuples with* `NID` *values of* 2 *and* 3 *in* `Neighborhood` *table as two condition instances to generate the tuples of* `Apartments` *located in these two neighborhoods respectively. Similar to the intra-table condition, we need to preserve the correlation between the condition and synthetic tuples, i.e., neighborhoods with a higher population density tend to have apartments with higher rent prices. Thus, the synthetic tuple with* `NID = 3` *tends to have rental prices greater than* $5K$. *Moreover, the synthetic tuples need to resemble the real data, and so the* `Accommodation` *of this synthetic tuple would have a large number.*

2.2 Diffusion Models: A Primer

Diffusion models [28] aim to learn a series of transitions to map a noise $\boldsymbol{z}$ sampled from a known prior distribution (usually Gaussian distribution) to a data $\boldsymbol{x}$ that follows an unknown target data distribution $p(\boldsymbol{x})$. To learn this process, it first defines a forward process to convert data distribution to noise distribution.

Forward Process. Given a data $\boldsymbol{x}$ sampled from the target data distribution $p(\boldsymbol{x})$ as the starting point $\boldsymbol{x}^0$, the forward process is defined by a series of Gaussian:

$$q(\boldsymbol{x}^t|\boldsymbol{x}^{t-1}) = \mathcal{N}(\boldsymbol{x}^t; \sqrt{1-\beta_t}\boldsymbol{x}^{t-1}, \beta_t \mathbf{I}), \tag{1}$$

which is a Markov process that can add a small amount of Gaussian noise to the $\boldsymbol{x}^0$ in T steps, producing a sequence of noisy data $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^T$. The variables $\{\beta_t \in (0,1)\}_{t=1}^T$ are called schedule

variables to control the variance of the noise in each step. We can express $\boldsymbol{x}^t$ at arbitrary timestep t with respect to $\boldsymbol{x}^0$ in a closed form:

$$q(\boldsymbol{x}^t|\boldsymbol{x}^0) = \mathcal{N}(\boldsymbol{x}^t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}^0, (1-\bar{\alpha}_t)\mathbf{I}), \tag{2}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=0}^{t} \alpha_t$. In this forward process, $\boldsymbol{x}^t$ will finally become a noise with Gaussian distribution when $t \to \infty$ [28].

Reverse Process. Contrary to the forward process, the reverse process $q(\boldsymbol{x}^{t-1}|\boldsymbol{x}^t)$ aims to gradually denoise a noisy data $\boldsymbol{x}^T$ to $\boldsymbol{x}^0$, where the denoised result $\boldsymbol{x}^0$ is the synthetic data with target data distribution. The reverse distributions $q(\boldsymbol{x}^{t-1}|\boldsymbol{x}^t)$ can be approximated as a Gaussian [28]:

$$p_\theta(\boldsymbol{x}^{t-1}|\boldsymbol{x}^t) = \mathcal{N}(\boldsymbol{x}^{t-1}; \mu_\theta(\boldsymbol{x}_t, t), \sigma_t^2\mathbf{I}), \tag{3}$$

where $\sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$ and $\mu_\theta(\boldsymbol{x}_t, t)$ can be derived from:

$$\mu_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}^t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)), \tag{4}$$

where $\boldsymbol{\epsilon}_\theta$ is a noise prediction model to estimate a predicted noise $\hat{\boldsymbol{\epsilon}}_t$ *w.r.t.* the standard Gaussian noise $\boldsymbol{\epsilon}_t$ injected to $\boldsymbol{x}^t$ in the forward process. The objective of the diffusion model is to obtain the noise prediction model $\boldsymbol{\epsilon}_\theta$, which can then be used to approximate the reverse distributions $q(\boldsymbol{x}^{t-1}|\boldsymbol{x}^t)$ for each reverse step.

3 SOLUTION OVERVIEW

As we have discussed previously, our goal is to generate desired synthetic tuples that capture both realism and correlation with an instance $\boldsymbol{c}$ of the given condition $\boldsymbol{C}$. To achieve this, we need to capture the conditional distribution $p(\boldsymbol{t}|\boldsymbol{c})$. Since the values in condition instance $\boldsymbol{c}$ are always pre-determined during the generation, by applying the Bayes' theorem, we can express the conditional distribution as follows:

$$p(\boldsymbol{t}|\boldsymbol{c}) = \frac{p(\boldsymbol{t})p(\boldsymbol{c}|\boldsymbol{t})}{p(\boldsymbol{c})} \propto p(\boldsymbol{t})p(\boldsymbol{c}|\boldsymbol{t}). \tag{5}$$

In this case, the conditional generation process that utilizes a generative model to directly capture $p(\boldsymbol{t}|\boldsymbol{c})$ and sample from the learned distribution can be converted to the following controllable process. We first use an unconditional generative model to capture $p(\boldsymbol{t})$, and then control this generative model to sample a synthetic tuple $\boldsymbol{t}'$ that can maximize the correlation satisfaction $p(\boldsymbol{c}|\boldsymbol{t})$ from the $p(\boldsymbol{t})$. To this end, we propose a diffusion-based framework RelDDPM consisting of two modules: (1) a *diffuser* module, which is a diffusion-based generative model used to capture the unconditional distribution $p(\boldsymbol{t})$. (2) a *controller* module, which is a function to approximate the correlation satisfaction $p(\boldsymbol{c}|\boldsymbol{t})$ to guide the diffuser during generation.

To be more specific, the controllable generation process using RelDDPM can be divided into two process as shown in Figure 3. Given a table $\boldsymbol{R}$ with distribution $p(\boldsymbol{t})$ and k different conditions $\boldsymbol{C}_1, \boldsymbol{C}_2, \ldots, \boldsymbol{C}_k$. During the training process, we first train a diffuser module `Diff` using the tuples in $\boldsymbol{R}$ to capture the unconditional distribution $p(\boldsymbol{t})$, which can preserve the realism of the synthetic tuple. Additionally, we train k controllers $\texttt{Ctrl}_1, \texttt{Ctrl}_2, \ldots, \texttt{Ctrl}_k$, where each controller corresponds to a specific condition. These controllers are designed to measure the correlation satisfaction of synthetic tuples according to their respective condition instances. In the sampling process, RelDDPM keeps the generative model, i.e., diffuser, frozen and steer its output according to different given condition instances by incorporating the corresponding controller in the generation process.

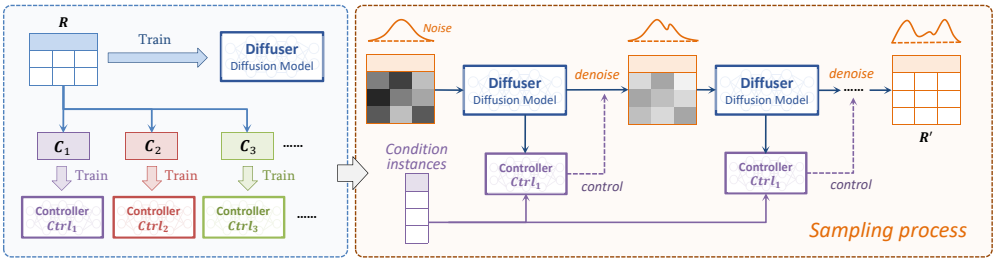


Fig. 3. An overview of the controllable table synthesis using RelDDPM.

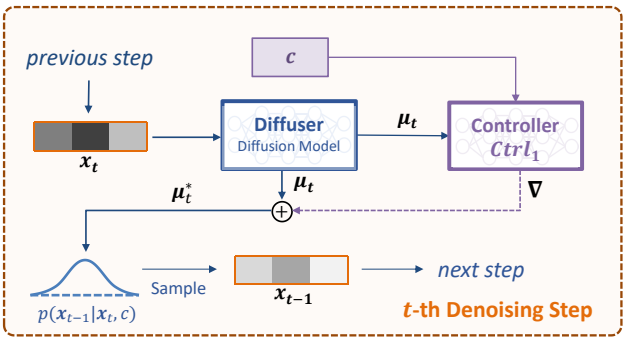


Fig. 4. Guided denoising step of RelDDPM.

Diffuser for Tabular Data Synthesis. The diffuser `Diff` is a diffusion model trained on $\boldsymbol{R}$ to capture its distribution. As we introduced in Section 2.2, `Diff` captures the target data distribution $p(\boldsymbol{x})$ by approximating a reverse process $p(\boldsymbol{x}^{t-1}|\boldsymbol{x}^t)$ to denoise a noisy data $\boldsymbol{x}^T \sim \mathcal{N}(0, \mathbf{I})$ to $\boldsymbol{x}^0 \sim p(\boldsymbol{x})$ in T timesteps. However, the diffusion model can not directly handle the tabular data as the state-of-the-art diffusion model operates in continuous space. To address this issue, we adopt the technique of *analog bit encoding* [12], which is an invertible data transformation method initially proposed for encoding discrete image data. By applying analog bit encoding we convert the tabular data into a continuous representation that can be handled by the diffuser. Further details on how we apply the data transformation and train the diffusion model specifically for tabular data can be found in Section 4.

Controller for Controllable Generation. We incorporate the corresponding controller `Ctrl` to measure the correlation satisfaction $p(\boldsymbol{c}|\boldsymbol{t})$ to control the diffuser `Diff` during the generation process, enabling controllable data synthesis. Specifically, a controller takes a tuple $\boldsymbol{t} \in \boldsymbol{R}$ and the condition instance $\boldsymbol{c}$ as the input to output a score that measures the probability that $\boldsymbol{t}$ satisfies the correlation with $\boldsymbol{c}$.

To illustrate this, let's consider the tables shown in Figure 1. Suppose we consider the attributes in table `Neighborhood` as the condition $\boldsymbol{C}$ and use the tuple $(NY, 26000)$ as a condition instance $\boldsymbol{c}$. We have two tuples $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ in `Apartment` table, where $\boldsymbol{t}_1 = (1, 1, 1, \leq 5K)$ and $\boldsymbol{t}_2 = (3, 5, 8, > 5K)$. Based on the joined result of these two tables, we can observe that apartments located in neighborhoods with high population density tend to have higher rent and accommodates. Therefore, we can determine that $\texttt{Ctrl}(\boldsymbol{t}_1, \boldsymbol{c}) < \texttt{Ctrl}(\boldsymbol{t}_2, \boldsymbol{c})$, indicating that $\boldsymbol{t}_2$ better satisfies the correlation defined $\boldsymbol{c}$

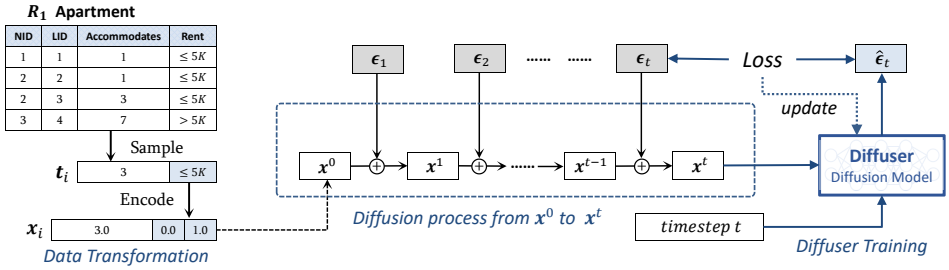


Fig. 5. Training of the diffuser for RelDDPM to capture the distribution of a given target table $\boldsymbol{R}$.

than $\boldsymbol{t}_1$. We can formulate our objective as an optimization objective with respect to `Ctrl`, aiming to generate a synthetic tuple $\boldsymbol{t}'$ with a higher value of $\texttt{Ctrl}(\boldsymbol{t}', \boldsymbol{c})$ as much as possible.

However, it is non-trivial to measure the correlation between relational tuples due to the high-dimensional and heterogeneous nature of the data, complex relational structures, and non-linear relationships, especially when we require this computation to be differentiable in order to utilize gradients as guidance. In this paper, we train two encoders for $\boldsymbol{t}'$ and $\boldsymbol{c}$ via contrastive learning and compute the similarity of the encoded tuples as the measurement. More details of how we design and train the controllers can be found in Section 5.

Combining the Diffuser and Controllers in Controllable Generation. After obtaining the diffuser `Diff` and controllers for each condition, we can now generate the corresponding synthetic tuples according to the given condition instances by combining the diffuser `Diff` and controller `Ctrl` in the generation process. As shown in Figure 3, the basic idea is to inject `Ctrl` at each denoising step of `Diff` to guide the `Diff` to generate a synthetic tuple that best fits the desired correlation according to the given condition instance. Specifically, given a condition instance $\boldsymbol{c}_i$ at the t-th denoising step, as shown in Figure 4, `Ctrl` guides the `Diff` by perturbing μ_t (the mean of the reverse distribution predicted by `Diff`) to $\mu_t^* = \mu_t + \sigma_t \cdot g_t$, where $g_t = \nabla_{\boldsymbol{x}^t}\texttt{Ctrl}(\boldsymbol{x}^t, \boldsymbol{c}_i)$, and then samples the $\boldsymbol{x}^{t-1}$ from the reverse distribution.

4 DIFFUSER CONSTRUCTION

Diffusion model [28], as a new kind of generative model emerging recently, has shown its extraordinary ability to approximate the underlying distributions of high-dimensional data in many data synthesis applications, and it can also be guided in the reverse process to generate the synthetic data conditionally [23, 44, 48]. Therefore, diffusion model is a natural fit for our framework. Figure 5 presents how we construct a diffusion model to capture the distribution of a given target table $\boldsymbol{R}$. We first transform the relational tuples into continuous vectors and then train the diffuser to capture the distribution of the transformed data.

4.1 Data Transformation

State-of-the-art diffusion models mainly operate in continuous space where forward and reverse processes are characterized by Gaussian distributions. Therefore, given a table $\boldsymbol{R}$ containing categorical attributes, we cannot directly use the diffusion model to capture the real data distribution $p(\boldsymbol{t})$. To this end, we employ an indirect approach to capture the original distribution. We transform each tuple $\boldsymbol{t}_i \in \boldsymbol{R}$ into a continuous vector $\boldsymbol{x}_i \in \mathbf{R}^d$, where d represents the dimension of the transformed tuple. To achieve this, we utilize an invertible encoding method called *analog bit encoding* [12] to convert the categorical variables in $\boldsymbol{t}_i$ into continuous variables. By applying this operation, we can map the original distribution $p(\boldsymbol{t})$, which encompasses mixed data types, into a continuous

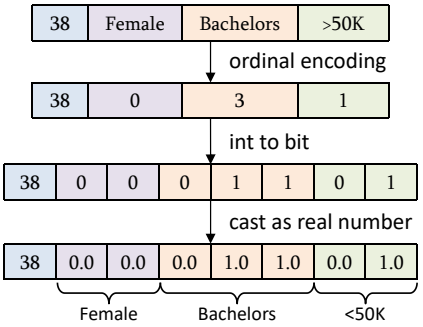


Fig. 6. Data transformation on categorical attributes.

distribution $p(\boldsymbol{x})$, which can then be captured by diffusion model. In the generation, we first sample the synthetic data $\boldsymbol{x}'$ from $p(\boldsymbol{x})$, then reverse it to the corresponding relational tuple $\boldsymbol{t}'$.

Analog Bit Encoding, which is proposed to encode the discrete image data [12], is used in our framework to preprocess categorical attributes for tabular data. Given a categorical attribute A_i, since A_i could contain strings that can not be directly processed by the neural network, we first assign an ordinal integer to each category of A_i, e.g., starting from 0 to $|A_i| - 1$, where $|A_i|$ is domain size of A_i, to transform A_i into a discrete numerical attribute. Afterward, we represent each ordinal integer of A_i using its corresponding binary bits, so each variable of the A_i will be converted to a binary vector, as $\{0, 1\}^d$, where $d = \lceil \log_2^{|A_i|} \rceil$. Then, we cast these binary bits $\{0, 1\}^d$ into real number $\mathbf{R}^d$ for continuous diffusion models. These real numbers are termed as *analog bits* as they learn to share the same bimodal values as binary bits but are modeled as real numbers. At generating time, the generated analog bits can be reversed to the binary bits by a thresholding operation, and then we can transform these binary bits to the original discrete numbers and decode the number to the corresponding values.

Combination of Multiple Attributes. Consider a table $\boldsymbol{R}$ with N_1 numerical attributes and N_2 categorical attributes. For each tuple $\boldsymbol{t} = \{\boldsymbol{t}_{num}, t_{cat_1}, t_{cat_2}, \ldots, t_{cat_{N_2}}\}$, which consists of numerical values $\boldsymbol{t}_{num} \in \mathbf{R}^{N_1}$ and N_2 categorical values $\{t_{cat_i}\}$. We transform each categorical value t_{cat_i} to the corresponding analog bit vector $\boldsymbol{x}_{cat_i} \in \mathbf{R}^{d_i}, d_i = \lceil \log_2^{|A_i|} \rceil$. Then, we concatenate all the above analog bit vectors with $\boldsymbol{t}_{num}$ to obtain the transformed vector $\boldsymbol{x} = \{\boldsymbol{t}_{num}, \boldsymbol{x}_{cat_1}, \boldsymbol{x}_{cat_2}, ..., \boldsymbol{x}_{cat_{N_2}}\}$ for the model training.

Example 2 (Data Transformation). *Let us consider the tuple in Figure 6. It consists of one numerical value and three categorical values. Before we feed it into the neural network, we transform the categorical values into the corresponding analog bits and obtain* (38, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0)*, where the underlines indicate different attributes.*

4.2 Diffusion Model Training

After data transformation, we can then formulate the forward process and reverse process for the tuples in $\boldsymbol{R}$. Given a transformed tuple $\boldsymbol{x}$, let $\boldsymbol{x}^t$ be the noisy data at the t-th diffusion step where $\boldsymbol{x}^0$ is the original data $\boldsymbol{x}$. As we introduced in Section 2.2, the forward diffusion process starting from $\boldsymbol{x}^0$ can be defined as:

$$q(\boldsymbol{x}^t|\boldsymbol{x}^{t-1}) = \mathcal{N}(\boldsymbol{x}^t; \sqrt{1-\beta_t}\boldsymbol{x}^{t-1}, \beta_t \mathbf{I}), \tag{6}$$

Algorithm 1: DiffuserTraining $(\boldsymbol{R}, T, \{\beta_t\}_{t=1}^T, N)$

Input: $\boldsymbol{R}$: Real table; T: timesteps of forward (reverse) process; $\{\beta_t\}_{t=1}^T$: schedule variables; N: number of training iterations
Output: The learned diffuser `Diff`

Initialize parameters θ of `Diff`
/* **Pre-compute the α_t and $\bar{\alpha}_t$ for all timesteps** */
$\{\alpha_t\}_{t=1}^T \leftarrow \{1-\beta_t\}_{t=1}^T$
$\{\bar{\alpha}_t\}_{t=1}^T \leftarrow \{\prod_{i=1}^{t} \alpha_i\}_{t=1}^T$
for *training epoch* $i = 1, 2, \ldots, N$ **do**
  Randomly sample $\boldsymbol{t} \in \boldsymbol{R}$
  Randomly sample t from `Uniform`$(\{1, ..., T\})$
  /* **Transform the tuple to continuous vector** */
  $\boldsymbol{x}^0 \leftarrow$ *data_transformation*$(\boldsymbol{t})$
  Sample noise $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathbf{I})$
  $\boldsymbol{x}^t \leftarrow \sqrt{\bar{\alpha}_t}\boldsymbol{x}^0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t$
  $\hat{\boldsymbol{\epsilon}}_t \leftarrow$ `Diff`$(\boldsymbol{x}^t, t)$
  $Loss \leftarrow ||\boldsymbol{\epsilon}_t - \hat{\boldsymbol{\epsilon}}_t||^2$
  Compute gradients $\nabla_\theta Loss$
  Update θ using stochastic gradient descent.
return `Diff`

where $\{\beta_t \in (0,1)\}_{t=1}^T$ are predetermined schedule variables. As t increases from 0 to T, the original data $\boldsymbol{x}^0$ will gradually become a Gaussian noise $\boldsymbol{x}^T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then we train a diffuser `Diff` to approximate the reverse process, i.e., denoising a Gaussian noise $\boldsymbol{x}^T$ to $\boldsymbol{x}^0$.

As shown in Figure 5, at each training iteration, when a tuple $\boldsymbol{t}$ is sampled from the target table $\boldsymbol{R}$, we need to first transform it into a continuous vector $\boldsymbol{x}$ and take it as $\boldsymbol{x}^0$, i.e., the starting point of the forward process. Then we sample a timestep t from $\{1, \ldots, T\}$ and obtain the corresponding noisy data $\boldsymbol{x}^t$ by adding the noise to $\boldsymbol{x}^0$. Afterward, we feed $\boldsymbol{x}^t$ and t into `Diff` to obtain $\hat{\boldsymbol{\epsilon}}_t$ i.e., the predicted noise added in the forward process, and update the parameters of `Diff` by minimizing the regression loss:

$$\min_\theta \mathbf{E}_{t, \boldsymbol{x}^0, \boldsymbol{\epsilon}_t} ||\boldsymbol{\epsilon}_t - \hat{\boldsymbol{\epsilon}}_t||^2, \tag{7}$$

where $\boldsymbol{\epsilon}_t$ is the real noise injected in the forward process. Algorithm 1 shows the pseudo-code of the diffuser training.

5 CONTROLLER CONSTRUCTION

In this section, we first conduct theoretical analyses on how the controller guides the diffuser module, followed by the design and training of different controllers for various conditions.

5.1 Theoretical Analysis on Controller

Given a condition $\boldsymbol{C}$, whether $\boldsymbol{C}$ is either intra-table or inter-table, generating the synthetic tuples that meet the condition instances of $\boldsymbol{C}$ essentially aims to capture the corresponding conditional probability distribution $p(\boldsymbol{t}|\boldsymbol{c})$ and sample the synthetic tuples from the learned distribution. Therefore, without loss of generality, we can consider that $\boldsymbol{C}$ is an intra-table condition in this section.

Given a condition instance $\boldsymbol{c}$, we denote the encoded $\boldsymbol{c}$ as $\boldsymbol{x}_c$. To control the diffuser to sample a synthetic data $\boldsymbol{x}'$ from conditional distribution $p(\boldsymbol{x}|\boldsymbol{x}_c)$, the reverse diffusion process at the t-th

denoising step becomes $q(\boldsymbol{x}^t|\boldsymbol{x}^{t+1}, \boldsymbol{x}_c)$, which is decomposed by

$$q(\boldsymbol{x}^t|\boldsymbol{x}^{t+1}, \boldsymbol{x}_c) \propto q(\boldsymbol{x}^t|\boldsymbol{x}^{t+1})p(\boldsymbol{x}_c|\boldsymbol{x}^t, \boldsymbol{x}^{t+1}), \tag{8}$$

where $q(\boldsymbol{x}^t|\boldsymbol{x}^{t+1})$ is the original reverse distribution, which can be then approximated by a learned Gaussian model $p_\theta(\boldsymbol{x}^t|\boldsymbol{x}^{t+1}) = \mathcal{N}(\boldsymbol{x}^t; \mu_\theta, \sigma_t^2\mathbf{I})$ (as mentioned in Section 2.2). For the second term $p(\boldsymbol{x}_c|\boldsymbol{x}^t, \boldsymbol{x}^{t+1})$, we can further simplify it to $p(\boldsymbol{x}_c|\boldsymbol{x}^t)$ via conditional independence assumption [42, 59]. Then, we have:

$$q(\boldsymbol{x}^t|\boldsymbol{x}^{t+1}, \boldsymbol{x}_c) \approx Z\mathcal{N}(\boldsymbol{x}^t; \mu_\theta, \sigma_t^2\mathbf{I})p(\boldsymbol{x}_c|\boldsymbol{x}^t), \tag{9}$$

where Z is a normalizing constant.

However, even though we have derived the form of the reverse distribution considering the condition instance $\boldsymbol{x}_c$, it is intractable to sample from this distribution directly. Fortunately, Sohl-Dickstein et al. [57] have proven that the distribution can be approximated as a perturbed Gaussian distribution. Specifically, we denote $\mathcal{N}(\boldsymbol{x}^t; \mu_\theta, \sigma_t^2\mathbf{I})$ as $\mathcal{N}(\mu, \sigma)$ for simplicity, and take the logarithm of both sides of Equation (9), i.e.,

$$\begin{aligned}
\log q(\boldsymbol{x}^t|\boldsymbol{x}^{t+1}, \boldsymbol{x}_c) &\approx \log[\mathcal{N}(\mu, \sigma)p(\boldsymbol{x}_c|\boldsymbol{x}^t)] \\
&= \log \mathcal{N}(\mu, \sigma) + \log p(\boldsymbol{x}_c|\boldsymbol{x}^t) \\
&\approx -\frac{1}{2}(\boldsymbol{x}^t - \mu)^T\sigma^{-1}(\boldsymbol{x}^t - \mu) + (\boldsymbol{x}^t - \mu)\cdot g + C \\
&= -\frac{1}{2}(\boldsymbol{x}^t - \mu - \sigma g)^T\sigma^{-1}(\boldsymbol{x}^t - \mu - \sigma g) + \frac{1}{2}g^T\sigma g + C \\
&= -\frac{1}{2}(\boldsymbol{x}^t - \mu - \sigma g)^T\sigma^{-1}(\boldsymbol{x}^t - \mu - \sigma g) + C \\
&= \log \mathcal{N}(\mu + \sigma\cdot g, \sigma),
\end{aligned} \tag{10}$$

where C is the constant term that can be ignored. After reverting $\mathcal{N}(\mu, \sigma)$ back to its original form, we can obtain:

$$q(\boldsymbol{x}^t|\boldsymbol{x}^{t+1}, \boldsymbol{x}_c) \approx Z\mathcal{N}(\boldsymbol{x}^t; \mu_\theta + \sigma_t\cdot g, \sigma_t^2\mathbf{I}), \tag{11}$$

where $g = \nabla_{\boldsymbol{x}^t}p(\boldsymbol{x}_c|\boldsymbol{x}^t)$. Sampling from the Gaussian distribution on the right-hand side of Equation (11) is straightforward. Therefore, as long as we can compute the gradient g, we can control the diffuser at each denoising step to sample from $q(\boldsymbol{x}^t|\boldsymbol{x}^{t+1}, \boldsymbol{x}_c)$ and eventually obtain the synthetic data following the conditional distribution $p(\boldsymbol{x}|\boldsymbol{x}_c)$.

Based on the above analysis, all we need to do is introduce a controller to estimate $p(\boldsymbol{x}_c|\boldsymbol{x}^t)$ and ensure that the calculation process is differentiable with respect to $\boldsymbol{x}^t$. As shown in Figure 4, when there comes a condition instance $\boldsymbol{c}$ at t-th denoising step, we can respectively obtain the approximated μ_θ from the diffuser and compute the gradient g from the controller, and then combine them to obtain the reverse distribution to sample the $\boldsymbol{x}_{t-1}$. We will discuss how to design such a controller in the next section.

5.2 Design of the Controller

As introduced above, we know that the diffusion model can generate data conditionally by injecting a controller to estimate $p(\boldsymbol{x}_c|\boldsymbol{x}^t)$, i.e., measuring the correlation satisfaction of $\boldsymbol{x}^t$ according to a condition instance $\boldsymbol{x}_c$ to guide the reverse sampling. We then present how to design our controller.

When the condition $\boldsymbol{C}$ contains only one attribute, such as a discrete class label y, the target for estimation becomes $p(y|\boldsymbol{x}^t)$, and we can train a classifier to predict the probability of $\boldsymbol{x}^t$ belonging to y as the controller [16]. When $\boldsymbol{C}$ is more meaningful and complex, such as an inter-table condition containing multiple attributes, we use the similarity between the embeddings of $\boldsymbol{x}_c$ and $\boldsymbol{x}^t$ to measure the correlation.

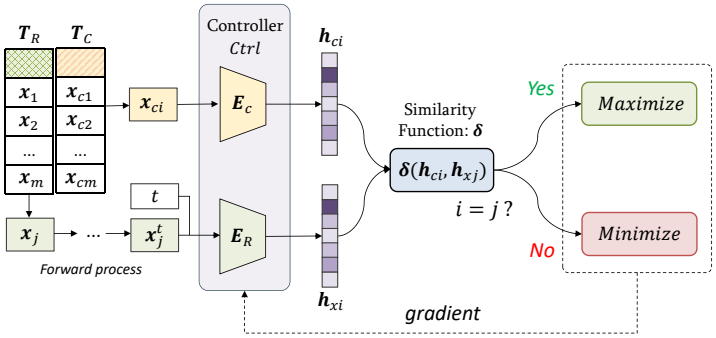


Fig. 7. Training the encoders E_R and E_c in the controller.

Let us take the related tables R_1 (Apartment) and R_2 (Neighborhood) in Figure 1 as example again. Suppose that we want to utilize the attributes in R_2 as an inter-table condition to generate the synthetic tuples for R_1. The controller Ctrl, which is used to measure the correlation between the tuples in R_1 and the condition instances, consists of two encoders E_c and E_R. Here, E_c is used for condition instances (tuples in R_2), and E_R is used for the tuples of the table to synthesize. Specifically, given the transformed tuples x and x_c, we first embed them into the joint embedding space using E_c and E_R respectively. Then we calculate the cosine similarity between the embeddings $E_R(x)$ and $E_c(x_c)$ as the similarity, which is used to formulate the correlation between x and x_c. Intuitively, the similarity between the $E_R(x)$ and $E_c(x_c)$ approximates the $\log p(x, x_c)$ [44]. The basic idea here is that if x and x_c are highly correlated, they are likely to co-occur in the real data. In this example, the correlated tuples would co-occur in the joined result $R_1 \bowtie R_2$, and thus the embeddings of them would also be very close in the latent space.

However, Ctrl is time-dependent as it needs to take the noisy data $x^0, x^1, ..., x^T$ in the reverse sampling process as the input. Therefore, the encoder E_R needs to incorporate the timestep t and the corresponding noisy data x^t as the input, i.e.,

$$\texttt{Ctrl}(x_c, x^t, t) = Cosine\_similarity(E_c(x_c), E_R(x^t, t)), \tag{12}$$

where $t \in [0, T]$ and x^t is the t-th noisy data sampled from the forward process $q(x^t|x^0)$.

5.3 Controller Training

Given a condition C and a table R, we train the encoders E_c and E_R in the controller Ctrl by optimizing the contrastive objective [52]. We first collect a training set $T_{train} = [T_R; T_C]$, where each $t_i \in T_R$ and $c_i \in T_C$ respectively represent real tuples and the corresponding condition instances. In the training process, we sample a batch of training tuples $\{t_i, c_i\}_{i=1}^m$ from the training set. For each c_i and t_i, we transform them into the continuous vectors x_{ci} and x_i via the data transformation method introduced in Section 4.1. Then we randomly sample a timestep t and obtain the t-th noisy data x_i^t and feed x_{ci} and x_i^t into the encoders to obtain the embeddings h_{ci} and h_{xi}. The contrastive objection aims to maximize the similarity of the embeddings of the N correlated tuples, i.e., $\sum_{i=1}^N similarity(h_{ci}, h_{xi})$ while minimizing the similarity of the embeddings of the $N^2 - N$ uncorrelated tuples, i.e., $\sum_{i \neq j} similarity(h_{ci}, h_{xj})$ (see Figure 7).

To this end, we optimize a symmetric cross-entropy loss over these similarities. We first compute a contrastive matrix $\mathbf{M} = (m_{ij}) \in \mathbf{R}^{N \times N}$, where m_{ij} is the similarity of h_{xi} and h_{cj}. The objective is to optimize the E_c and E_R to make $\mathbf{M}$ close to an identity matrix as much as possible. This training

Algorithm 2: ControllerTraining $(\boldsymbol{T}_{train}, m, N, T, \{\beta_t\}_{t=1}^{T})$

Input: $\boldsymbol{T}_{train}$: training set; m: batch size; N: number of training iterations; T: timesteps of the reverse process; $\{\beta_t\}_{t=1}^{T}$: schedule variables
Output: The learned controller $\texttt{Ctrl} = \{\boldsymbol{E}_c, \boldsymbol{E}_R\}$

Initialize parameters ϕ_c of encoder $\boldsymbol{E}_c$
Initialize parameters ϕ_R of encoder $\boldsymbol{E}_R$
/* **Pre-compute the** α_t **and** $\bar{\alpha}_t$ **for all timesteps** */
$\{\alpha_t\}_{t=1}^{T} \leftarrow \{1-\beta_t\}_{t=1}^{T}$
$\{\bar{\alpha}_t\}_{t=1}^{T} \leftarrow \{\prod_{i=1}^{t} \alpha_i\}_{t=1}^{T}$
for *training iteration* $n = 1, 2, \ldots, N$ **do**
  Randomly sample m training data $\{\boldsymbol{t}_i, \boldsymbol{c}_i\}_{i=1}^{m}$ from $\boldsymbol{T}_{train}$
  Randomly sample t from $\texttt{Uniform}(\{1, ..., T\})$
  for $i = 1, 2, \ldots, m$ **do**
    $\boldsymbol{x}_{ci} \leftarrow data\_transformation(\boldsymbol{c}_i)$
    $\boldsymbol{x}_i^0 \leftarrow data\_transformation(\boldsymbol{t}_i)$
    sample $\boldsymbol{x}_i^t \sim q(\boldsymbol{x}^t|\boldsymbol{x}^0) = \mathcal{N}(\boldsymbol{x}^t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}^0, (1-\bar{\alpha}_t)\mathbf{I})$
    $\boldsymbol{h}_{ci} \leftarrow E_c(\boldsymbol{x}_{ci})$
    $\boldsymbol{h}_{xi} \leftarrow E_R(\boldsymbol{x}_i^t, t)$
  $\mathbf{M} \leftarrow (\boldsymbol{h}_{c1}, \boldsymbol{h}_{c2}, ..., \boldsymbol{h}_{cm}) \cdot (\boldsymbol{h}_{x1}, \boldsymbol{h}_{x2}, ..., \boldsymbol{h}_{xm})^{\top}$
  $Loss\_1 \leftarrow \frac{1}{m}\sum_{L=1}^{m} \text{cross\_entropy\_loss}(\mathbf{M}[L, :], L)$
  $Loss\_2 \leftarrow \frac{1}{m}\sum_{L=1}^{m} \text{cross\_entropy\_loss}(\mathbf{M}[:, L], L)$
  $Loss \leftarrow Loss\_1 + Loss\_2$
  Compute gradients $\nabla_{\phi_c} Loss$ and $\nabla_{\phi_R} Loss$
  Update ϕ_c and ϕ_R using stochastic gradient descent.
$\texttt{Ctrl} \leftarrow \{\boldsymbol{E}_c, \boldsymbol{E}_R\}$
return Ctrl

technique and objective are first introduced as multi-class N-pair objective in [58], and are recently used in contrastive representation learning [41, 58]. Algorithm 2 presents the pseudo-code.

5.4 Controllable Synthesis

Given a target table $\boldsymbol{R}$ and a set of instances $\boldsymbol{T}_C$ of the condition $\boldsymbol{C}$, RelDDPM utilizes the controller Ctrl to guide the diffuser Diff in the reverse process to generate the synthetic table $\boldsymbol{R}'$ that follows the conditional distribution $P(\boldsymbol{t}|\boldsymbol{c})$.

Algorithm 3 presents the process of conditional table synthesis. For each condition instance $\boldsymbol{c}_i \in \boldsymbol{T}_C$, we first encode it to the continuous vector $\boldsymbol{x}_{ci}$. In the reverse process, we utilize Diff to obtain the mean and variance of $p(\boldsymbol{x}^{t-1}|\boldsymbol{x}^t)$ (Lines 8-10). Then we estimate the $p(\boldsymbol{x}_{ci}|\boldsymbol{x}^t)$ via Ctrl and compute the gradient to obtain the mean of the conditional reverse process $p(\boldsymbol{x}^{t-1}|\boldsymbol{x}^t, \boldsymbol{x}_{ci})$ (Lines 11-12), where the scaling factor s is a user-controlled hyper-parameter that determines the strength of the guidance. Afterwards, we sample the $\boldsymbol{x}^{t-1}$ from the $p(\boldsymbol{x}^{t-1}|\boldsymbol{x}^t, \boldsymbol{x}_{ci})$ via reparameterized trick (Lines 13-14). The sampling process can be accelerated by launching multiple threads on GPUs.

6 EXPERIMENTS

We conduct experiments to evaluate the performance of RelDDPM. Section 6.2 and Section 6.3 respectively evaluate the two core modules: *diffuser* and *controller* to answer the following questions:

Algorithm 3: CondSynthesis (T_C, `Diff`, `Ctrl`, s, T, $\{\beta_t\}_{t=1}^T$).

Input: T_C: condition instances; `Diff`: trained diffuser; `Ctrl`: trained controller; s: scaling factor; T: timesteps of the reverse process; $\{\beta_t\}_{t=1}^T$: schedule variables;
Output: Generated table R'

/* **Pre-compute the α_t and $\bar{\alpha}_t$ for all timesteps** */
$\{\alpha_t\}_{t=1}^T \leftarrow \{1-\beta_t\}_{t=1}^T$
$\{\bar{\alpha}_t\}_{t=1}^T \leftarrow \{\prod_{i=1}^{t} \alpha_i\}_{t=1}^T$
for $i = 1, 2, \ldots, |T_C|$ **do**
  /* **Encode the condition tuple to continuous vector** */
  $\boldsymbol{x}_{ci} \leftarrow data\_transformation(\boldsymbol{c}_i)$
  Randomly sample $\boldsymbol{x}^T \sim \mathcal{N}(0, \mathbf{I})$
  for $t = T, \ldots, 1$ **do**
    $\hat{\boldsymbol{\epsilon}}_t \leftarrow \texttt{Diff}(\boldsymbol{x}^t, t)$
    $\mu_t \leftarrow \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}^t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\boldsymbol{\epsilon}}_t)$
    $\sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
    $g_t \leftarrow \nabla_{\boldsymbol{x}^t}\texttt{Ctrl}(\boldsymbol{x}_{ci}, \boldsymbol{x}^t, t)$
    $\mu_*^c \leftarrow \mu_t + s \cdot g_t$
    Randomly sample $\boldsymbol{z} \sim \mathcal{N}(0, \mathbf{I})$
    $\boldsymbol{x}^{t-1} \leftarrow \mu_t^* + \sigma_t^2 \cdot \boldsymbol{z}$
  /* **Decode $\boldsymbol{x}^0$ to a relational tuple** */
  $\boldsymbol{t}' \leftarrow data\_transformation(\boldsymbol{x}^0)$
  Append $\boldsymbol{t}'$ to $\boldsymbol{R}'$
return R'

• **Q1:** Is the diffuser module effective in capturing the unconditional data distribution for tabular data?
• **Q2:** Is the controller module effective in capturing the correlation between conditions and synthetic results, and correctly guiding the diffuser during the generation process?

Next, we evaluate the performance of controllable data synthesis in RelDDPM through two downstream applications: *minority class oversampling* in Section 6.4 and *missing tuple completion* in Section 6.5 to answer the following questions:
• **Q3:** Is RelDDPM effective in handling intra-table conditions?
• **Q4:** Is RelDDPM effective in handling inter-table conditions?

6.1 Experimental Setup

Datasets. To comprehensively evaluate the quality of synthetic results generated by RelDDPM, we utilize 7 real-world datasets in our experiments. Specifically, we utilize 3 single table datasets: *(1) Default* [2] contains information on credit card clients, including demographic factors, credit data, and payment default status. It is a single table with 30,000 tuples and 24 attributes. It is commonly used for predicting if a client would default on their credit card payments. The dataset is imbalanced, with the majority class accounting for 77.9% and the minority class accounting for 22.1%.
(2) Shoppers [55] focuses on online shoppers' behavior and contains information about their browsing and purchasing activities. It is a single table with 12,330 tuples and 18 attributes. It is often

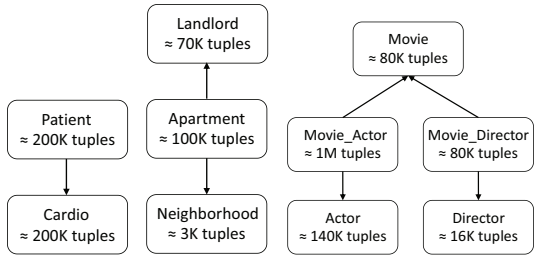


Fig. 8. Real-world datasets that consist of multiple tables.

used for predicting shoppers' purchasing intentions. The dataset is imbalanced, with the majority class accounting for 84.5% of the tuples and the minority class accounting for 15.5%.
(3) WeatherAUS [5] is a collection of daily weather observations from various weather stations in Australia spanning a period of 10 years. The target variable we focus on for classification is "RainTomorrow", which indicates whether or not it rains on a given day. The dataset is imbalanced, with the majority (minority) class accounting for 78.0% (22.0%) of the tuples.

Moreover, we also select 3 real-world databases that consist of multiple tables:
(4) Heart [3] is a heart disease database that contains two tables: `Patient` and `Cardio`. The `Patient` table includes patient information, while the `Cardio` table contains 8 electrocardiogram indices. These two tables can be joined using the patient ID, and the joined result has 14 attributes and 200, 000 tuples.
(5) Airbnb [1] is originally a single table of housing information derived from Airbnb database. We follow [27] to normalize it to obtain three different tables: `Landlord`, `Neighborhood` and `Apartment`. The joined result of these three tables contains 26 attributes and 104, 072 tuples.
(6) Imdb [4] is derived from the popular IMDB database. We normalize it to obtain five tables. Specifically, we merge the tables about movies to obtain the `Movie` table and divide the original person table into `Director` and `Actor`, which can be joined with the `Movie` table via `Movie_Actor` and `Movie_Director` respectively.

Figure 8 shows the schemas of these three datasets.

Baselines. We compare RelDDPM with 6 state-of-the-art tabular data synthesis methods, covering 3 mainstream generative models: Generative Adversarial Networks (GAN), Auto-regressive Models (ARM) and Diffusion Models. Specifically, we select 3 methods that focus on capturing unconditional distribution for tabular data:
(1) OCTGAN [34] is a GAN-based tabular data synthesis model. It improves the quality of synthetic data by designing the generator and discriminator based on neural ordinary differential equations.
(2) TabDDPM [37] is a diffusion-based tabular data synthesis framework. Unlike our diffuser module, It handles categorical attributes by incorporating a multinomial diffusion process.
(3) SAM [65] is an ARM-based tabular data synthesis framework. It concentrates on synthesizing the database consisting of multiple tables to support both single table synthesis and joined table synthesis. We only compare our approach with SAM in terms of single table synthesis.

Besides that, we also compare RelDDPM with the following three conditional tabular data synthesis frameworks to examine if our controllable synthesis framework can outperform these existing methods in handling different conditions:
(4) SOS [36] is a score-based synthesis framework. It is proposed to oversample the tuples with minority class in the original data by considering the label as an *intra-table* condition, and generate synthetic samples accordingly.

(5) Restore [27] utilizes auto-regressive models (ARMs) to complete the incomplete joined results. It uses ARM to capture the conditional table distribution from the observed joined tuples and generate the missing tuples in the joined result using observed tuples as *inter-table conditions*.
(6) CTGAN [64] is a framework that utilizes conditional GAN to capture the tabular data distribution. It incorporates an additional condition input during training and sampling to handle the multi-modal continuous attributes and imbalanced categorical attributes.

Implementation Details. We use an MLP architecture to implement the diffuser modules in RelDDPM:

$$\texttt{MLP}(\boldsymbol{x}_{input}) = \texttt{Linear}(\texttt{MLPBlock}(\ldots(\texttt{MLPBlock}(\boldsymbol{x}_{input}))))$$
$$\texttt{MLPBlock}(\boldsymbol{x}_{input}) = \texttt{Dropout}(\texttt{ReLU}(\texttt{Linear}(\boldsymbol{x}_{input})))$$

To incorporate the timestep t, we follow [16, 49] to use a sinusoidal time embedding to obtain the input by:

$$t_{emb} = \texttt{Linear}(\texttt{SiLU}(\texttt{Linear}(\texttt{SinTimeEmb}(t))))$$
$$\boldsymbol{x}_{input} = \texttt{Linear}(\boldsymbol{x}^t) + t_{emb},$$

where SinTimeEmb represents the sinusoidal time embedding layer. Specifically, we use a 4-hidden-layers MLP with neuron numbers [512, 1024, 1024, 512] to implement the diffuser. We also implement the encoders in the controller module with this MLP architecture. For each encoder, we use a 2-layers MLP with neuron numbers [512, 512] to implement it. We implement all the neural networks using PyTorch.

All experiments are conducted on a Ubuntu 16.04 server with 62TB disk, 40 CPU cores (Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz), one GPU (NVIDIA GeForce RTX 3090) and 1TB memory, and the version of Python is 3.8.8.

6.2 Evaluation on Diffuser Module

This section evaluates the effectiveness of the diffuser module in RelDDPM in capturing the unconditional distribution for tabular data. To this end, we utilize the three single table datasets, namely Default, Shoppers and WeatherAUS, and compare RelDDPM with the three unconditional table synthesis methods: OCTGAN, TabDDPM and SAM.

Given a real table $\boldsymbol{R}$, we use an unconditional generative model, including the diffuser in RelDDPM and other baselines, to generate a synthetic table $\boldsymbol{R'}$. Then, we evaluate the performance of the model by measuring if $\boldsymbol{R'}$ is "similar" to $\boldsymbol{R}$ from the following aspects.

Evaluation on Distribution Differences. We first directly evaluate if the $\boldsymbol{R'}$ has a similar distribution with $\boldsymbol{R}$. To this end, we follow [65] to compute the cross entropy for each individual attribute of $\boldsymbol{R'}$ with regard to that of $\boldsymbol{R}$. We report the sum of the entropy for numerical attributes and categorical attributes in Table 1. From the results, we can find that RelDDPM excels at resembling both numerical and categorical attributes. For SAM, it can achieve comparable performance with RelDDPM on categorical attributes sometimes. However, SAM fails to resemble the numerical attributes well as the ARM model used in SAM can not handle the attributes with a large domain size well [26, 66]. For TabDDPM, it can achieve a satisfied performance on numerical attributes, but the performance on categorical attributes is inferior to our method. In the case of OCTGAN, due to the instability of the adversarial training process in GAN-based methods, the model may not easily achieve optimal status, resulting in its performance being the worst among all the baselines.

Evaluation on Data Utility Differences. Besides the direct evaluation on data distribution, we also follow the existing works [17, 37, 65] to evaluate the *utility* of the synthetic data $\boldsymbol{R'}$ to evaluate whether the generative models capture joint distribution of the real data. Specifically, we evaluate the utility of $\boldsymbol{R'}$ in two different applications, i.e., ML model training and OLAP querying answering.

Table 1. Evaluation on distribution differences (i.e., cross entropy) of the synthetic data w.r.t real data.

Dataset	Attributes	Approaches			
		RelDDPM	TabDDPM	SAM	OCTGAN
Default	**Categorical**	**0.018**	0.030	0.019	0.111
	Numerical	0.290	**0.287**	0.751	1.906
Shoppers	**Categorical**	**0.023**	0.151	0.100	0.151
	Numerical	**0.840**	0.861	0.849	3.528
WeatherAUS	**Categorical**	**0.003**	0.383	0.010	0.379
	Numerical	**0.287**	0.528	0.738	5.115

Table 2. Evaluation on data utility differences of the synthetic data w.r.t. real data.

Metric	Dataset	RelDDPM	TabDDPM	SAM	OCTGAN
F1 Diff	Default	**0.001**	0.014	0.016	0.032
	Shoppers	**0.007**	0.076	0.054	0.059
	WeatherAUS	**0.020**	0.043	0.036	0.051
RE	Default	**0.149**	0.255	0.321	0.386
	Shoppers	**0.182**	0.429	0.295	1.133
	WeatherAUS	**0.187**	0.515	0.268	0.331

(1) Data Utility for ML Model Training. Given the real table $\boldsymbol{R}$ and the synthetic table $\boldsymbol{R'}$, we hope an ML model f' trained on the synthetic table can achieve a similar performance as the model f trained on the real table. In this case, we adopt the *F1 difference* [17] as the metric to measure the data utility of $\boldsymbol{R'}$ with respect to $\boldsymbol{R}$, which is defined as: $\textit{F1 difference} = |F1(f'|\boldsymbol{R}_{test}) - F1(f|\boldsymbol{R}_{test})|$, where $\boldsymbol{R}_{test}$ is a test data used to measure the F1 scores for f and f'. The lower the F1 difference, the less performance gap between f and f', i.e., the better data utility of $\boldsymbol{R'}$.
(2) Data Utility for OLAP Query Answering. Given the real table $\boldsymbol{R}$ and the synthetic table $\boldsymbol{R'}$, we examine the utility of $\boldsymbol{R'}$ by using it to answer OLAP queries and evaluate the accuracy of the query results on $\boldsymbol{R'}$ with respect to $\boldsymbol{R}$. In this case, we compute the *Relative Error* of $q(\boldsymbol{R'})$ with respect to $q(\boldsymbol{R})$ to measure the utility of $\boldsymbol{R'}$. The lower the relative error is, the better data utility of $\boldsymbol{R'}$ is.

Table 2 provides a comparison of the diffuser in RelDDPM with other tabular data synthesis frameworks in preserving the data utility. We report the results on MLP classifier in ML model training for simplicity. We can find that the results are consistent with the previous evaluation results on individual attribute distributions: RelDDPM still outperforms the other methods. This indicates that the diffuser module, combined with the analog bit encoding technique, is effective in capturing tabular data distribution.

6.3 Evaluation on Controller Module

This section evaluates if our controller module in RelDDPM can effectively capture the correlation between conditions and synthetic results, and correctly guide the diffuser to generate synthetic data that meet specified conditions based on the captured correlation.

To this end, we utilize the three datasets that consist of multiple tables: `Heart`, `Airbnb` and `Imdb`. Specifically, for dataset `Heart`, we use `Patient` as $\boldsymbol{R}_c$ and `Cardio` table as $\boldsymbol{R}$. For dataset `Airbnb`, we use `Landlord` table as $\boldsymbol{R}_c$ and `Apartment` table as $\boldsymbol{R}$. For dataset `Imdb`, we use `Actor` table as $\boldsymbol{R}_c$ and `Movie` table as $\boldsymbol{R}$. we compare RelDDPM with two methods, `Restore` and `CTGAN`.

Given a target table $\boldsymbol{R}$ and a condition table $\boldsymbol{R}_c$ that has a join relationship with $\boldsymbol{R}$, we first obtain the join result $\boldsymbol{R}_c \bowtie \boldsymbol{R}$ and calculate a correlation matrix M based on $\boldsymbol{R}_c \bowtie \boldsymbol{R}$, with the (i, j)-th element in M representing the correlation coefficient between the i-th attribute in $\boldsymbol{R}$ and the j-th attribute in $\boldsymbol{R}_c$. We follow [37] to use the Pearson correlation coefficient for numerical-numerical

Table 3. Evaluation on correlation differences of the synthetic data w.r.t real data.

Dataset	Approaches		
	RelDDPM	ReStore	CTGAN
Heart	**0.0028**	0.0103	0.0158
Airbnb	**0.0082**	0.0102	0.0249
Imdb	**0.0085**	0.0098	0.0819

attributes, the correlation Ratio for numerical-categorical attributes, and Theil's U statistic between categorical attributes. Afterward, we use the attributes in $\boldsymbol{R}_c$ as an inter-table condition and the tuples in $\boldsymbol{R}_c$ as the condition instances to generate corresponding synthetic joined result $\boldsymbol{R}_c \bowtie \boldsymbol{R}'$. Then, we also compute a correlation matrix M' based on the synthetic result. We then compute the absolute difference between the M and M' as an evaluation. A smaller difference between the matrices indicates that the correlation between the condition instances and the synthetic tuples in $\boldsymbol{R}_c \bowtie \boldsymbol{R}'$ is closer to real-world scenarios. This suggests that the synthesis method used to generate $\boldsymbol{R}'$ effectively captures and maintains the correlations present in the real data.

We report the average of the absolute difference between correlation matrices computed on real and synthetic data in Table 3. We observe that RelDDPM achieves the minimal absolute difference in most cases. This indicates that the controller module in RelDDPM can effectively capture the correlation between the conditions and the target data in real scenarios, and guide the diffuser to generate the synthetic data that satisfies the given conditions based on the learned correlation.

6.4 Evaluation on Minority Class Oversampling

To evaluate the performance of RelDDPM in performing controllable data synthesis under intra-table conditions, we focus on the task of minority class oversampling, which is commonly used in machine learning, especially classification task, to address imbalanced label distribution in the original training data. The objective of minority class oversampling is to generate synthetic samples that balance the label distribution of the training data, thereby improving the model's performance on the minority class.

In this evaluation, we utilize RelDDPM to oversample the minority class by treating the label as an intra-table condition and controlling the generative model to generate the tuples belonging to the minority class. The performance of RelDDPM can be assessed by evaluating the effectiveness of oversampling, such as by measuring the performance of the classification model trained on the training data after oversampling.

Evaluation Framework. Given a table $\boldsymbol{R}$, we first randomly split the original table into a training set $\boldsymbol{R}_{train}$ and a test set $\boldsymbol{R}_{test}$ using a $4:1$ ratio. For the training set $\boldsymbol{R}_{train}$, we apply different oversampling approaches, including RelDDPM and the baseline methods, to generate synthetic tuples for the minority class. These synthetic tuples are added to the original training set $\boldsymbol{R}_{train}$ to create an augmented training set $\boldsymbol{R}'_{train}$, which has a balanced label distribution after oversampling.

Next, we use the augmented training set $\boldsymbol{R}'_{train}$ to train a classifier f. The classifier is then evaluated on the test set $\boldsymbol{R}_{test}$ using the F1 score as the performance metric. To comprehensively evaluate the oversampling performance, we consider the four types of traditional machine learning classifiers: Decision Tree (DT), Random Forest (RF), AdaBoost (AB), Logical Regression (LR), and one deep learning classifier Multilayer Perceptron (MLP). We focus on the F1 score on the minority class, a higher F1 score indicates a better performance of the minority class oversampling technique.

Datasets & Baselines. We use the 3 single table datasets: `Default`, `Shoppers` and `WeatherAUS` as they all have skew label distribution. We compare the performance of RelDDPM with two generative-based baseline approaches: `SOS`, which is proposed to address the minority class oversampling task,

Table 4. Comparison on minority class oversampling.

Dataset	Classfier	Approaches				
		Identity	SMOTE	CTGAN	SOS	RelDDPM
Default	**DT**	0.406	0.402	0.386	0.410	**0.455**
	RF	0.419	0.508	0.473	0.511	**0.535**
	AB	0.444	0.519	0.471	0.506	**0.532**
	LR	0.455	0.515	0.504	0.459	**0.531**
	MLP	0.484	0.531	0.469	0.465	**0.541**
Shoppers	**DT**	0.544	0.555	0.561	0.555	**0.568**
	RF	0.582	**0.641**	0.631	0.620	0.637
	AB	0.598	0.627	0.634	0.627	**0.640**
	LR	0.495	0.602	0.409	0.573	**0.668**
	MLP	0.602	0.592	0.439	0.611	**0.616**
WeatherAUS	**DT**	0.551	0.550	0.540	0.567	**0.581**
	RF	0.604	0.616	0.595	0.613	**0.646**
	AB	0.611	0.638	0.579	0.633	**0.642**
	LR	0.633	0.642	0.561	0.635	**0.659**
	MLP	0.626	0.633	0.610	**0.661**	0.654

and CTGAN, which can be easily adapted to this task. Moreover, we also compare our method with SMOTE [11], which is a traditional method for minority class oversampling.

Evaluation Results. Table 4 presents the evaluation results of using RelDDPM and other approaches in the task of minority class oversampling. Identity represents the results obtained by training the classifier on the original training data without oversampling. From the results, we can observe that RelDDPM consistently performs well on the three datasets and five classifiers. With a few exceptions, such as the RF classifier trained on Shoppers and the MLP classifier trained on the WeatherAUS dataset, RelDDPM outperforms the other approaches. This indicates the effectiveness of our controllable data synthesis framework in addressing the imbalanced label distribution through intra-table conditions.

Among the baseline approaches, CTGAN performs the worst, which is consistent with the observations from the previous experimental results. The unstable adversarial training strategies of GAN-based methods can lead to the generation of noisy data. As a result, the augmented training data obtained by CTGAN sometimes hampers the classifier's performance compared to the Identity results. SMOTE and SOS achieve comparable performance, which is worse than that of RelDDPM. SMOTE may generate tuples that violate the attribute correlation in the original data, as it interpolates between neighboring samples. Thus, SMOTE may not be well-suited for tabular datasets with categorical attributes, as direct interpolation of neighboring categorical values can introduce unreasonable values. For SOS, as the score-based model is originally used for capturing the distribution on continuous space, the approach may fail to generate the tabular data with categorical data well. Additionally, training a conditional generative model from scratch can be challenging to achieve the same data quality as an unconditional generative model [14, 51].

Overall, the results demonstrate that the RelDDPM framework that utilizes the controllers to guide the diffuser to generate the desired data is effective for controllable data synthesis with intra-table conditions.

6.5 Evaluation on Missing Tuple Completion

Missing tuple completion aims to generate synthetic tuples to complete the missing fields in an incomplete joined result caused by systematic missing [27], which is a typical application of controllable data synthesis under inter-table condition.

As the example illustrated in Figure 1. Suppose we have a query that involves the joined result Apartment $\bowtie$ Neighborhood. However, the joined result is incomplete due to the absence of tuples

Table 5. Missing tuple injection.

Dataset	Biased Attribute	Setup	Corrupted Tables	Keep Rates
Heart	Cardio.oldpeak	H_1	Cardio	40% ~ 80%
		H_2	Patient	60%
			Cardio	40% ~ 80%
Airbnb	Apartment.price	A_1	Apartment	40% ~ 80%
		A_2	Landlord	60%
			Apartment	40% ~ 80%
Imdb	Movie.rating	I_1	Movie	40% ~ 80%
		I_2	Movie_Actor	60%
			Movie_Director	60%
			Movie	40% ~ 80%

for apartments located in the fourth neighborhood, which could introduce bias in subsequent analysis or computations. In this case, we can use the tuples in the Neighborhood table as inter-table condition instances to generate the corresponding join partners for the tuples that failed to be joined.

Datasets & Baselines. We utilize the 3 datasets that consist of multiple tables: Heart, Airbnb and Imdb, and compare RelDDPM with two state-of-the-art conditional tabular data synthesis methods that can be used to handle the inter-table conditions, ReStore and CTGAN.

Evaluation Framework. To evaluate the performance of controllable data synthesis under inter-table conditions, we first compute the original joined results in Heart, Airbnb and Imdb datasets. Then, we manually corrupt the tables in the joined results to inject the missing tuples and derive the incomplete joined results for them. Next, we utilize different approaches to "restore" the missing tuples based on the correlations with complete tuples, and evaluate their performance by comparing with the original join results.

To this end, given a complete joined table, we follow [27] to first select a particular attribute A and use the values of A to determine if a given tuple should be removed. Specifically, we control the Pearson correlation coefficient between the probability of a tuple being removed with the value of attribute A to be 1, and thus the values in missing tuples would highly depend on the value of A. This can make remaining tuples be out of distribution. Then, we choose the tables to corrupt, i.e., deciding which table's tuple will be removed. We use a parameter *keep rate* to determine the percentage of tuples which are not removed from the original joined result. To comprehensively evaluate our method, we define two setups for each dataset. Specifically, as depicted in Table 5, we consider removing tuples in only one table (i.e., setups H_1, A_1 and I_1 for the three datasets), and vary the keep rate from 40% to 80%. On the other hand, we also consider removing tuples from more tables (i.e., setups H_2, A_2 and I_2) to simulate more complex missing cases, while controlling keep rate for the tuples of these tables to be 60%.

After manually creating the incomplete joined results, we can apply different approaches to synthesize the missing tuples and obtain the complete joined results. To evaluate the synthetic quality, we use the *relative error reduction* metric [27], which is defined based on an aggregation query q that is executed on both the incomplete joined result J^i and the completed joined result J^c. The goal is to measure how much the relative error of the query result can be reduced by completing the joined result. We first calculate the query result $q(J^i)$ on the incomplete joined result and $q(J^c)$ on the completed joined result. Then, we compute the relative errors of these query results with respect to the ground truth result $q(J)$, the relative error reduction can be computed by:

$$Rel.Error\ Reduction = Er(q(J^i), q(J)) - Er(q(J^c), q(J)). \tag{13}$$

For each incomplete joined result, we prepare 1000 queries and compute the averages of the error reduction to comprehensively evaluate the performance.

Besides the *relative error reduction*, we also consider using *bias reduction* to measure the performance of missing tuple completion. The idea of this metric is to evaluate how well the real complete data distribution of a given attribute can be restored [27]. Given a continuous attribute A, the bias reduction is defined as:

$$Bias\ Reduction = 1 - \frac{|AVG^c(A) - AVG(A)|}{|AVG(A) - AVG^i(A)|} \tag{14}$$

where $AVG^c(A), AVG^i(A), AVG(A)$ denotes the averages of attribute A on J^c, J^i and J respectively. Different from the *relative error reduction*, *bias reduction* can show the performance of missing tuple completion independent of a given workload.

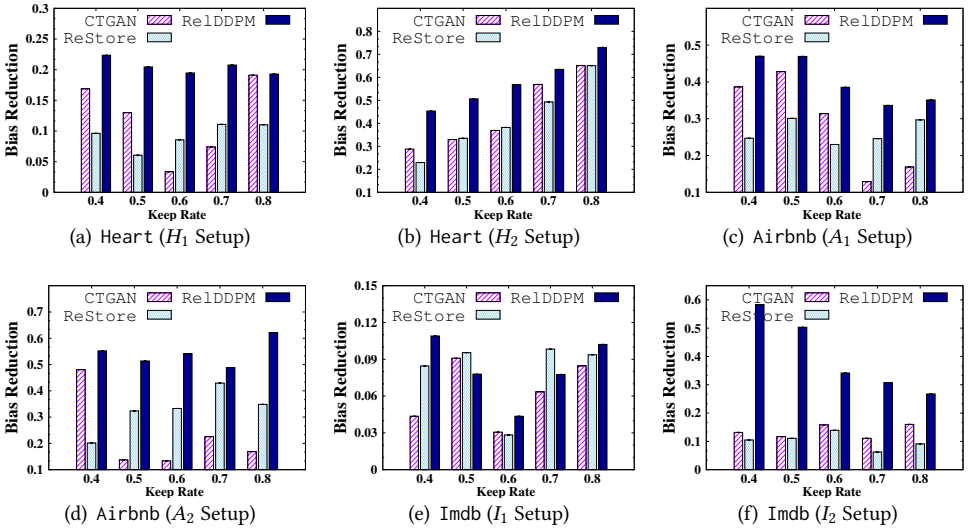
Evaluation Results. We first evaluate the effectiveness of RelDDPM by measuring the bias reduction and relative error reduction, and then evaluate the efficiency of RelDDPM by using the processing time of completing the incomplete joined results.

(1) Evaluation on Bias Reduction. From the results reported in Figure 9, we observe that RelDDPM outperforms both `CTGAN` and `ReStore` on all the three datasets, indicating the effectiveness of our controller module in capturing the correlation between the biased attribute with the conditions and guiding the diffuser to generate the correct value of biased attribute for the missing tuples.

Specifically, `CTGAN` performs the worst, as the adversarial training process is not stable, making it difficult to obtain an optimized generator. `ReStore` performs better than `CTGAN`, but it still cannot surpass the performance of RelDDPM. Furthermore, we find that the bias reduction achieved by RelDDPM is significantly higher than that of `ReStore`, particularly when the keep rate is low. For example, as shown in Figures 9(a) and 9(b), on the H_1 and H_2 setups of the `Heart` dataset with a keep rate of 0.4, RelDDPM achieves a bias reduction that is 132% and 97% higher than `ReStore`, respectively. This improvement is mainly attributed to the fact that `ReStore` can only utilize the completed joined tuples to train the generative model, as it needs to capture the conditional distribution. In contrast, RelDDPM can use all tuples when training the unconditional generative model. Thus, the diffuser in RelDDPM would perform better than the conditional generative model in `ReStore`.

(2) Evaluation on Error Reduction. We compare our approach with other methods of answering aggregation queries. Figure 10 reports the evaluation on the setups of the `Heart`, `Airbnb` and `Imdb` datasets. We find that RelDDPM consistently outperforms `CTGAN` and `ReStore`, which indicates that the controller in RelDDPM preserves the complex correlations across the different tables and guides the diffuser to generate the corresponding missing tuples to answer the aggregation queries correctly.

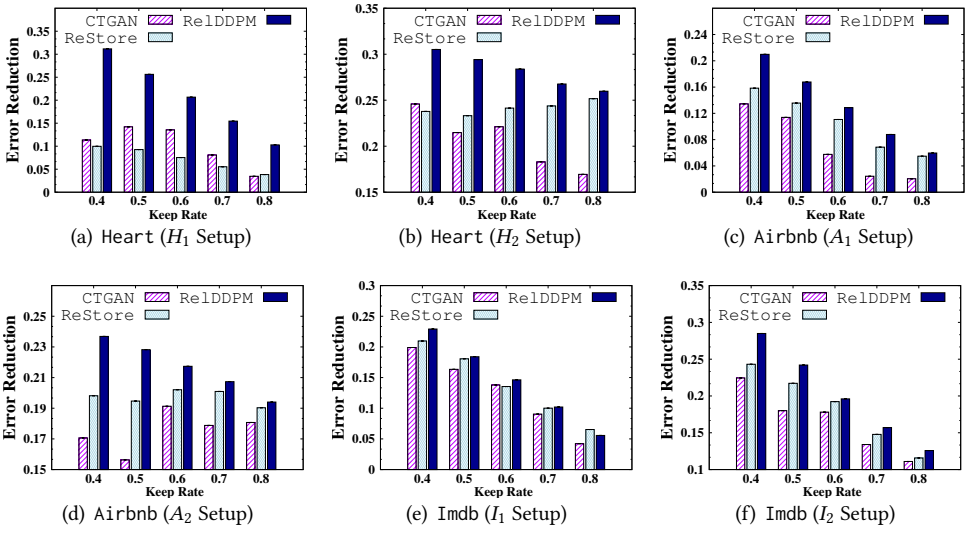
(3) Evaluation on the Efficiency. We also evaluate the efficiency of our method by comparing the processing time of RelDDPM with the baseline approaches. The processing time includes both the training time and the sampling time (measured in seconds). From the results reported in Tables 6, 7 and 8, we observe that the processing time of RelDDPM is significantly lower than that of `CTGAN` and `ReStore`. This is mainly because `CTGAN` and `ReStore` require training separate conditional generative models for each incomplete joined result. For instance, in the setup I_1 of the `Imdb` dataset, they need to train two generative models for the incomplete joined results `Actor` $\bowtie$ `Movie` and `Director` $\bowtie$ `Movie`. In contrast, RelDDPM only needs to train a single diffuser for the `Movie` table with two controllers that guide the diffuser to generate the movie tuples based on the `Actor` and `Director` tuples, respectively. Compared to the generative models, the training cost of the controller in RelDDPM is significantly lower. As a result, the majority of the time savings come from the training phase of the model, leading to faster processing time.



(a) Heart (H_1 Setup) (b) Heart (H_2 Setup) (c) Airbnb (A_1 Setup)

(d) Airbnb (A_2 Setup) (e) Imdb (I_1 Setup) (f) Imdb (I_2 Setup)

Fig. 9. Evaluating the *Bias Reduction* on Heart, Airbnb and Imdb datasets.



(a) Heart (H_1 Setup) (b) Heart (H_2 Setup) (c) Airbnb (A_1 Setup)

(d) Airbnb (A_2 Setup) (e) Imdb (I_1 Setup) (f) Imdb (I_2 Setup)

Fig. 10. Evaluating the *Error Reduction* on Heart, Airbnb and Imdb datasets.

Table 6. Comparison of the Processing Time Cost over the Different Approaches (Heart Dataset).

Setup	Methods	Processing Time		
		Training Time (s)	Sampling Time (s)	All Time (s)
H_1	CTGAN	1771.94	4.78	1776.72
	ReStore	1229.83	81.25	1311.08
	RelDDPM	**26.45**	**3.08**	**29.53**
H_2	CTGAN	2247.79	6.44	2254.22
	ReStore	507.36	162.12	198.52
	RelDDPM	**32.48**	**3.92**	**36.4**

Table 7. Comparison of the Processing Time Cost over the Different Approaches (Airbnb Dataset).

Setup	Methods	Processing Time		
		Training Time (s)	Sampling Time (s)	All Time (s)
A_1	CTGAN	1114.71	9.13	1123.84
	ReStore	189.11	**2.23**	191.34
	RelDDPM	**28.64**	3.69	**32.33**
A_2	CTGAN	1470.24	11.84	1482.08
	ReStore	254.63	**1.07**	255.7
	RelDDPM	**40.1**	4.91	**45.01**

Table 8. Comparison of the Processing Time Cost over the Different Approaches (Imdb Dataset).

Setup	Methods	Processing Time		
		Training Time (s)	Sampling Time (s)	All Time (s)
I_1	CTGAN	1673.22	7.49	1680.71
	ReStore	180.01	**3.39**	183.4
	RelDDPM	**22.04**	15.26	**37.3**
I_2	CTGAN	1362.45	9.61	1372.06
	ReStore	135.84	**4.33**	140.17
	RelDDPM	**21.49**	17.28	**38.77**

7 RELATED WORK

Synthetic Data Generation aims to create a synthetic dataset that can closely mirror the distribution of a given real dataset. For image generation, deep generative models (DGM) have shown significant performance in capturing image data distribution. Radford et al. [53] propose DCGAN, a GAN-based framework implemented by convolutional and deconvolutional networks to better capture the structural information of images. This work has catalyzed a series of GAN-based image generation research endeavors [19, 33, 56, 68, 72]. In the domain of text generation, there also have been numerous works based on DGM, such as GANs [24, 39, 67], and the highly popular large language models [8, 15, 60] that have gained significant traction recently.

Compared to image or textual data, the characteristics of heterogeneous data types, join relationships between tables, and correlations among attributes all pose challenges when addressing tabular data generation using deep generative models, which induce a series of research efforts in tabular data synthesis based on deep generative models.

Tabular Data Synthesis has been extensively studied, particularly with the advent of methods based on deep generative models. Some studies [13, 17, 34, 38, 43, 50, 61, 64] consider using generative adversarial networks (GAN) [22] to synthesize tabular data. They propose various preprocessing techniques and model design strategies to capture the distribution of tabular data. Yang et al. [65] and Hilprecht et al. [27] adopt auto-regressive models (ARM) [21], and they approach the task of tabular data synthesis as a sequence generation problem, generating the synthetic table attribute by attribute. Moreover, diffusion-based generative models also have been investigated in tabular data synthesis [28, 35–37] to achieve more stable synthetic quality.

However, most of these studies only concentrate on capturing the unconditional data distribution for tabular data. They focus on how to design and train the generative models to improve the quality of the synthetic data without considering any conditions, while the objective of ours is how to flexibly and efficiently control the generative models to generate the synthetic data according to different given conditions.

Controllable Data Generation aims to fix a given generative model while making it to support generating data conditionally according to different conditions. Some studies [6, 25, 31] propose to control the output of GAN by discovering the meaningful directions in the latent space of the generative model. Dhariwal et al. [16] propose to utilize the classifiers to control the language

models to generate the text that satisfies desired requirements. Liu et al. [44] and Graikos et al. [23] also utilize classifiers to control the category of the synthetic images. Liu et al. [44] devise a semantic controllable framework to support using text or images to control the synthetic images.

However, the proposed controllable generation methods are only used in image and text generation. Firstly, the generative models used for image and text generation cannot be readily applied to generate tabular data. Additionally, the methods designed to measure the correlation between generation conditions and images or text are not easily translated to tabular data. Thus, the existing controllable data synthesis frameworks can not be directly applied to tabular data. There is still a lack of exploration on controllable tabular data synthesis, which is the main technique discussed in this paper.

Privacy-Preserving Data Synthesis. The existing techniques proposed to preserve the privacy of the original data using synthetic data can be classified into two categories [29]: (1) *Privacy-preserving data publishing* [10, 20, 46, 71] that aims to anonymize the original data. The related approaches remove or replace personal identity information or sensitive data in the original dataset with unidentifiable data, thereby reducing the risk of privacy leakage. (2) *Privacy-preserving deep generative models* [9, 18, 32, 63, 69] that aim to train a generative model to generate the synthetic data while preserving privacy. These techniques consider introducing various privacy-preserving mechanisms (such as differential privacy) during the process of model training and data sampling to protect the privacy of the original data.

However, existing works have found that it is hard to effectively preserve the realism of the synthetic data while theoretically preserving the privacy for the original data.

8 CONCLUSION & FUTURE DIRECTIONS

In this paper, we have proposed a novel framework, RelDDPM, that is capable of synthesizing tabular data under different conditions, which could either be intra-table or inter-table. RelDDPM first utilizes diffusion models to capture unconditional data distribution, which serves as the foundation for generating tabular data without specific conditions. Building upon this foundation, we proposed a lightweight controller tailored for synthesizing tables with different conditions. The results of the experiments have validated the efficacy of the proposed approach. The combined use of diffusion models and the lightweight controller enables the generation of diverse tables that satisfy different conditions while maintaining the realism of the synthetic data. The flexibility and controllability of the approach make it suitable for various applications.

Moreover, privacy-preserving is a crucial factor to consider in data generation, particularly in scenarios related to data sharing and dissemination. We plan to conduct a thorough study in the future work, including the addition of evaluations for privacy-preserving within the evaluation framework. We will also investigate methods for preserving the privacy of the original data when conditions are involved during the generation process, and how to strike a balance between synthetic quality and the effectiveness of privacy-preserving.

ACKNOWLEDGMENTS

This work was partly supported by the NSF of China (62122090, 62072461, 61925205, 62232009, 62102215 and 62072458), National Key R&D Program of China (2023YFB4503600), the Fund for Building World-Class Universities (Disciplines) of Renmin University of China, the Beijing Natural Science Foundation (L222006), the Research Funds of Renmin University of China, and the Beijing National Research Center for Information Science and Technology (BNRist).

REFERENCES

[1] [n. d.]. Airbnb Data Set. https://public.opendatasoft.com/explore/dataset/airbnb-listings.

[2] [n. d.]. Default Data Set. https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients.

[3] [n. d.]. Heart Data Set. https://www.openml.org/data/download/6358/BNG_heart-statlog.arff.

[4] [n. d.]. Imdb Data Set. http://homepages.cwi.nl/~boncz/job/imdb.tgz.

[5] [n. d.]. WeatherAUS Data Set. https://www.kaggle.com/jsphyg/weather-dataset-rattle-package.

[6] Rameen Abdal, Peihao Zhu, Niloy J. Mitra, and Peter Wonka. 2021. StyleFlow: Attribute-conditioned Exploration of StyleGAN-Generated Images using Conditional Continuous Normalizing Flows. *ACM Trans. Graph.* 40, 3 (2021), 21:1–21:21. https://doi.org/10.1145/3447648

[7] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. 2021. Deep Neural Networks and Tabular Data: A Survey. *CoRR* abs/2110.01889 (2021). arXiv:2110.01889 https://arxiv.org/abs/2110.01889

[8] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[9] Zhipeng Cai, Zuobin Xiong, Honghui Xu, Peng Wang, Wei Li, and Yi Pan. 2021. Generative adversarial networks: A survey toward private and secure applications. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–38.

[10] Tânia Carvalho, Nuno Moniz, Pedro Faria, and Luís Antunes. 2022. Survey on Privacy-Preserving Techniques for Data Publishing. *CoRR* abs/2201.08120 (2022). arXiv:2201.08120 https://arxiv.org/abs/2201.08120

[11] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.* 16 (2002), 321–357. https://doi.org/10.1613/jair.953

[12] Ting Chen, Ruixiang Zhang, and Geoffrey E. Hinton. 2022. Analog Bits: Generating Discrete Data using Diffusion Models with Self-Conditioning. *CoRR* abs/2208.04202 (2022). https://doi.org/10.48550/arXiv.2208.04202 arXiv:2208.04202

[13] Edward Choi, Siddharth Biswal, Bradley A. Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. 2017. Generating Multi-label Discrete Patient Records using Generative Adversarial Networks. In *Proceedings of the Machine Learning for Health Care Conference, MLHC 2017, Boston, Massachusetts, USA, 18-19 August 2017 (Proceedings of Machine Learning Research, Vol. 68)*, Finale Doshi-Velez, Jim Fackler, David C. Kale, Rajesh Ranganath, Byron C. Wallace, and Jenna Wiens (Eds.). PMLR, 286–305. http://proceedings.mlr.press/v68/choi17a.html

[14] Yunjey Choi, Min-Je Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. 2018. StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 8789–8797. https://doi.org/10.1109/CVPR.2018.00916

[15] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling Language Modeling with Pathways. *CoRR* abs/2204.02311 (2022). https://doi.org/10.48550/arXiv.2204.02311 arXiv:2204.02311

[16] Prafulla Dhariwal and Alexander Quinn Nichol. 2021. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 8780–8794. https://proceedings.neurips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html

[17] Ju Fan, Tongyu Liu, Guoliang Li, Junyou Chen, Yuwei Shen, and Xiaoyong Du. 2020. Relational Data Synthesis using Generative Adversarial Networks: A Design Space Exploration. *Proc. VLDB Endow.* 13, 11 (2020), 1962–1975. http://www.vldb.org/pvldb/vol13/p1962-fan.pdf

[18] Liyue Fan. 2020. A survey of differentially private generative adversarial networks. In *The AAAI Workshop on Privacy-Preserving Artificial Intelligence*, Vol. 8.

[19] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. 2018. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing* 321 (2018), 321–331. https://doi.org/10.1016/j.neucom.2018.09.013

[20] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. 2010. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.* 42, 4 (2010), 14:1–14:53. https://doi.org/10.1145/1749603.1749605

[21] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. 2015. MADE: Masked Autoencoder for Distribution Estimation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015 (JMLR Workshop and Conference Proceedings, Vol. 37)*, Francis R. Bach and David M. Blei (Eds.). JMLR.org, 881–889. http://proceedings.mlr.press/v37/germain15.html

[22] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (Eds.). 2672–2680. https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html

[23] Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. 2022. Diffusion models as plug-and-play priors. *CoRR* abs/2206.09012 (2022). https://doi.org/10.48550/arXiv.2206.09012 arXiv:2206.09012

[24] Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long Text Generation via Adversarial Training with Leaked Information. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 5141–5148. https://doi.org/10.1609/aaai.v32i1.11957

[25] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. 2020. GANSpace: Discovering Interpretable GAN Controls. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/6fe43269967adbb64ec6149852b5cc3e-Abstract.html

[26] Shohedul Hasan, Saravanan Thirumuruganathan, Jees Augustine, Nick Koudas, and Gautam Das. 2020. Deep Learning Models for Selectivity Estimation of Multi-Attribute Queries. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 1035–1050. https://doi.org/10.1145/3318464.3389741

[27] Benjamin Hilprecht and Carsten Binnig. 2021. ReStore - Neural Data Completion for Relational Databases. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 710–722. https://doi.org/10.1145/3448016.3457264

[28] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

[29] Yuzheng Hu, Fan Wu, Qinbin Li, Yunhui Long, Gonzalo Munilla Garrido, Chang Ge, Bolin Ding, David A. Forsyth, Bo Li, and Dawn Song. 2023. SoK: Privacy-Preserving Data Synthesis. *CoRR* abs/2307.02106 (2023). https://doi.org/10.48550/arXiv.2307.02106 arXiv:2307.02106

[30] Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. 2023. LayoutDM: Discrete Diffusion Model for Controllable Layout Generation. *CoRR* abs/2303.08137 (2023). https://doi.org/10.48550/arXiv.2303.08137 arXiv:2303.08137

[31] Ali Jahanian, Lucy Chai, and Phillip Isola. 2020. On the "steerability" of generative adversarial networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=HylsTT4FvB

[32] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. 2019. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=S1zk9iRqF7

[33] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=Hk99zCeAb

[34] Jayoung Kim, Jinsung Jeon, Jaehoon Lee, Jihyeon Hyeong, and Noseong Park. 2021. OCT-GAN: Neural ODE-based Conditional Tabular GANs. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 1506–1515. https://doi.org/10.1145/3442381.3449999

[35] Jayoung Kim, Chaejeong Lee, and Noseong Park. 2022. STaSy: Score-based Tabular data Synthesis. *CoRR* abs/2210.04018 (2022). https://doi.org/10.48550/arXiv.2210.04018 arXiv:2210.04018

[36] Jayoung Kim, Chaejeong Lee, Yehjin Shin, Sewon Park, Minjung Kim, Noseong Park, and Jihoon Cho. 2022. SOS: Score-based Oversampling for Tabular Data. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, Aidong Zhang and Huzefa Rangwala (Eds.). ACM, 762–772. https://doi.org/10.1145/3534678.3539454

[37] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. 2022. TabDDPM: Modelling Tabular Data with Diffusion Models. *CoRR* abs/2209.15421 (2022). https://doi.org/10.48550/arXiv.2209.15421 arXiv:2209.15421

[38] Jaehoon Lee, Jihyeon Hyeong, Jinsung Jeon, Noseong Park, and Jihoon Cho. 2021. Invertible Tabular GANs: Killing Two Birds with One Stone for Tabular Data Synthesis. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 4263–4273. https://proceedings.neurips.cc/paper/2021/hash/22456f4b545572855c766df5eefc9832-Abstract.html

[39] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep Reinforcement Learning for Dialogue Generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, Jian Su, Xavier Carreras, and Kevin Duh (Eds.). The Association for Computational Linguistics, 1192–1202. https://doi.org/10.18653/v1/d16-1127

[40] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. 2022. Diffusion-LM Improves Controllable Text Generation. In *NeurIPS*. http://papers.nips.cc/paper_files/paper/2022/hash/1be5bc25d50895ee656b8c2d9eb89d6a-Abstract-Conference.html

[41] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020. Oscar: Object-Semantics Aligned Pre-training for Vision-Language Tasks. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXX (Lecture Notes in Computer Science, Vol. 12375)*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer, 121–137. https://doi.org/10.1007/978-3-030-58577-8_8

[42] Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. 2022. Diffusion-LM Improves Controllable Text Generation. *CoRR* abs/2205.14217 (2022). https://doi.org/10.48550/arXiv.2205.14217 arXiv:2205.14217

[43] Tongyu Liu, Ju Fan, Yinqing Luo, Nan Tang, Guoliang Li, and Xiaoyong Du. 2021. Adaptive Data Augmentation for Supervised Learning over Missing Data. *Proc. VLDB Endow.* 14, 7 (2021), 1202–1214. https://doi.org/10.14778/3450980.3450989

[44] Xihui Liu, Dong Huk Park, Samaneh Azadi, Gong Zhang, Arman Chopikyan, Yuxiao Hu, Humphrey Shi, Anna Rohrbach, and Trevor Darrell. 2021. More Control for Free! Image Synthesis with Semantic Diffusion Guidance. *CoRR* abs/2112.05744 (2021). arXiv:2112.05744 https://arxiv.org/abs/2112.05744

[45] Chao Ma, Sebastian Tschiatschek, Richard Turner, José Miguel Hernández-Lobato, and Cheng Zhang. 2020. VAEM: a deep generative model for heterogeneous mixed type data. *Advances in Neural Information Processing Systems* 33 (2020), 11237–11247.

[46] Abdul Majeed and Sungchang Lee. 2021. Anonymization Techniques for Privacy Preserving Data Publishing: A Comprehensive Survey. *IEEE Access* 9 (2021), 8512–8545. https://doi.org/10.1109/ACCESS.2020.3045700

[47] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. *CoRR* abs/1411.1784 (2014). arXiv:1411.1784 http://arxiv.org/abs/1411.1784

[48] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. 2017. Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 3510–3520. https://doi.org/10.1109/CVPR.2017.374

[49] Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved Denoising Diffusion Probabilistic Models. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8162–8171. http://proceedings.mlr.press/v139/nichol21a.html

[50] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. 2018. Data Synthesis based on Generative Adversarial Networks. *Proc. VLDB Endow.* 11, 10 (2018), 1071–1083. https://doi.org/10.14778/3231751.3231757

[51] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. Semantic Image Synthesis With Spatially-Adaptive Normalization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA,*

USA, June 16-20, 2019. Computer Vision Foundation / IEEE, 2337–2346. https://doi.org/10.1109/CVPR.2019.00244

[52] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8748–8763. http://proceedings.mlr.press/v139/radford21a.html

[53] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1511.06434

[54] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*. PMLR, 1278–1286.

[55] Cemal Okan Sakar, Suleyman Olcay Polat, Mete Katircioglu, and Yomi Kastro. 2019. Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks. *Neural Comput. Appl.* 31, 10 (2019), 6893–6908. https://doi.org/10.1007/s00521-018-3523-0

[56] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. 2019. SinGAN: Learning a Generative Model From a Single Natural Image. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 4569–4579. https://doi.org/10.1109/ICCV.2019.00467

[57] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015 (JMLR Workshop and Conference Proceedings, Vol. 37)*, Francis R. Bach and David M. Blei (Eds.). JMLR.org, 2256–2265. http://proceedings.mlr.press/v37/sohl-dickstein15.html

[58] Kihyuk Sohn. 2016. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 1849–1857. https://proceedings.neurips.cc/paper/2016/hash/6b180037abbebea991d8b1232f8a8ca9-Abstract.html

[59] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-Based Generative Modeling through Stochastic Differential Equations. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=PxTIG12RRHS

[60] Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Samuel Madden, and Mourad Ouzzani. 2021. RPT: Relational Pre-trained Transformer Is Almost All You Need towards Democratizing Data Preparation. *Proc. VLDB Endow.* 14, 8 (2021), 1254–1261. https://doi.org/10.14778/3457390.3457391

[61] Boris van Breugel, Trent Kyono, Jeroen Berrevoets, and Mihaela van der Schaar. 2021. DECAF: Generating Fair Synthetic Data Using Causally-Aware Generative Networks. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 22221–22233. https://proceedings.neurips.cc/paper/2021/hash/ba9fab001f67381e56e410575874d967-Abstract.html

[62] L Vivek Harsha Vardhan and Stanley Kok. 2020. Generating privacy-preserving synthetic tabular data using oblivious variational autoencoders. In *Proceedings of the Workshop on Economics of Privacy and Data Labor at the 37 th International Conference on Machine Learning (ICML)*.

[63] Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. 2018. Differentially Private Generative Adversarial Network. *CoRR* abs/1802.06739 (2018). arXiv:1802.06739 http://arxiv.org/abs/1802.06739

[64] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling Tabular data using Conditional GAN. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 7333–7343. https://proceedings.neurips.cc/paper/2019/hash/254ed7d2de3b23ab10936522dd547b78-Abstract.html

[65] Jingyi Yang, Peizhi Wu, Gao Cong, Tieying Zhang, and Xiao He. 2022. SAM: Database Generation from Query Workloads with Supervised Autoregressive Models. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 1542–1555. https://doi.org/10.1145/3514221.3526168

[66] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M. Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Deep Unsupervised Cardinality Estimation. *Proc. VLDB Endow.* 13, 3 (2019), 279–292. https://doi.org/10.14778/3368289.3368294

[67] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco,*

California, USA, Satinder Singh and Shaul Markovitch (Eds.). AAAI Press, 2852–2858. https://doi.org/10.1609/aaai.v31i1.10804

[68] Han Zhang, Tao Xu, and Hongsheng Li. 2017. StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 5908–5916. https://doi.org/10.1109/ICCV.2017.629

[69] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2014. PrivBayes: private data release via bayesian networks. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, Curtis E. Dyreson, Feifei Li, and M. Tamer Özsu (Eds.). ACM, 1423–1434. https://doi.org/10.1145/2588555.2588573

[70] Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. 2022. Guided Conditional Diffusion for Controllable Traffic Simulation. *CoRR* abs/2210.17366 (2022). https://doi.org/10.48550/arXiv.2210.17366 arXiv:2210.17366

[71] Bin Zhou, Jian Pei, and Wo-Shun Luk. 2008. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explor.* 10, 2 (2008), 12–22. https://doi.org/10.1145/1540276.1540279

[72] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2242–2251. https://doi.org/10.1109/ICCV.2017.244

Received July 2023; revised October 2023; accepted November 2023