End-to-End Federated Diffusion Generative Models for Tabular Data

by

Jiaming Xu

to obtain the degree of Master of Science
at Delft University of Technology,
to be defended publicly on Thursday June 22, 2023.

Student Number:	5247578
Programme:	Computer Science
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science
Thesis Committee:	Dr. Lydia Chen, TU Delft, Supervisor
	Hans Brower, BlueGen, Supervisor
	Dr. Gosia Migut, TU Delft

TU Delft

Preface

This report provides a summary of the thesis work conducted at the final part of the Master of Science in Computer Science program at Delft University of Technology. I am delighted to have the opportunity to conduct research on tabular data synthesis as my thesis topic.

Tabular data refers to structured data that is organized in a table-like format, the utilization of tabular data is pervasive across diverse domains and applications, making the synthesis of such data an active area of research. Especially in scenarios where the availability of real-world data is restricted, the utilization of synthesized data holds significant promise.

Recently, Diffusion Generative Models have demonstrated significant improvements in image synthesis. However, their effectiveness in synthesizing tables is limited due to the use of One-Hot encoding for representing categorical attributes with many categories. Additionally, these models require the private data to be centrally collected for training, which violates the criteria for privacy preservation. Besides improvement over representation of categorical features, one important aspect of this research is the development of methods for privacy-preserving data synthesis, which aims to generate synthetic data that retains the statistical properties while protecting the privacy of individuals in the dataset.

In general, we propose a new decentralized tabular probabilistic synthesizing framework, it has three key features: (i) decentralized Autoencondoer comprised of an encoder and a decoder to map discrete features into the continuous space and back, (ii) a tabular diffusion model trained in a decentralized manner and (iii) incorporating differential privacy on local stochastic gradient training. To assess our method, we conduct extensive experimental studies that focus on sampling quality and diversity, using 9 tabular datasets and 4 state-of-the-art synthesizers.

This master thesis has been finished under the supervision of Prof. Lydia Chen and Hans Brower. I extend my sincere appreciation to them, as their invaluable guidance and support have been essential throughout the completion of this project. Furthermore, I would like to express my gratitude to my family and friends for their consistent support.

Jiaming Xu
Delft, June 2023

Contents

1

Research Paper

End-to-End Federated Diffusion Generative Models for Tabular Data

Abstract—Tabular data is widely used in various fields and applications, making the synthesis of such data an active area of research. One important aspect of this research is the development of methods for privacy-preserving data synthesis, which aims to generate synthetic data that retains statistical properties while protecting the privacy of individuals in the dataset. Recently, Diffusion Generative Models, such as Gaussian Diffusion Model, have significantly improved image synthesis, but their effectiveness in synthesizing tables is limited, because of using One-Hot encoding for representing categorical attributes with many categories. Furthermore, it needs the private data to be centrally collected for training, thus violating the privacy-preserving criteria. In this paper, we propose a new decentralized tabular synthesizing framework, AutoDiffuse, which has three key features: (i) a decentralized Autoencoder comprised of an encoder and a decoder to map discrete features into the continuous space and back, (ii) a tabular diffusion model trained in a decentralized manner and (iii) incorporating differential privacy on central stochastic gradient training. We conduct extensive experimental studies that focus on sampling quality and diversity, using 9 tabular datasets and 4 state-of-the-art synthesizers. The results show that our method outperforms existing central methods by 10.7% and 31.4% in data quality and diversity on average, and 6.8% and 21.1% in data quality and diversity in scenarios facing non-IID data.

Index Terms—Tabular Data Synthesizer, Diffusion Generative Model, Federated Learning.

1 Introduction

Tabular data, which is organized into rows and columns, consisting of categorical and numerical values, is prevalent in a wide range of fields and applications. It is commonly used to represent structured data, such as medical records [1], survey responses [2], financial transactions [3], and many other types of information. The widespread use of tabular data is largely due to its versatility and ease of understanding, as it can be easily manipulated, analyzed, and visualized.

Data synthesis is a valuable tool for a variety of purposes, including data augmentation, privacy-preserving data mining. While the generation of images, audio and text has been widely studied, the generation of synthetic tabular data remains a challenge due to the mixed structure of discrete and continuous features, as well as their varying data distributions. Recently, the breakthrough in image generation with Diffusion Generative Models, such as Stable Diffusion [4], has drawn a lot of attention. Score-based Generative Models [5], a more general form of Diffusion Generative Models, have two processes, one is Diffusion process or Forward SDE, the other is Reverse process or Reverse SDE. Despite its great performance on image generation tasks, the possibility of it being a tabular data synthesizer hasn't been fully studied yet.

Recent studies, such as TabDDPM [6], STaSy [7], attempt to adopt Diffusion Generative Models to synthesize tabular table but they only consider centralized learning environment and do not consider changing the representation of data, particularly categorical variables, to improve performance. Since tabular data has the property of having hybrid types of features, which are categorical features and numerical features, to send this type of data into a Neural Network, the first step is to convert all categorical features into numerical values. Typically, there are two ways for doing this. The first one is using preprocessing techniques like One-Hot encoder or Ordinal encoder. But they both have their own limitations. For the One-Hot encoder, it usually leads to bad scalability, which means the discrete values will be represented as 0 and 1 vectors, and the table will be expended as much as the number of categories. The second one, Ordinal encoder, has the problem of incorporating redundant information, as some of the original categories don't have any orders, but will still be encoded into consecutive numbers.

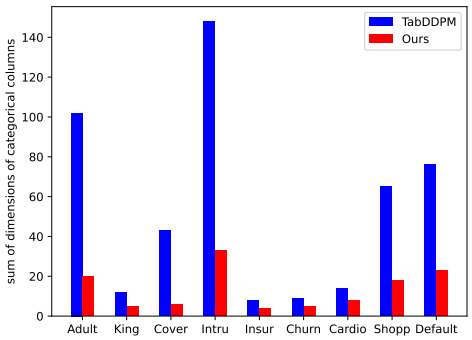

Fig. 1. The sum of the dimensions of all categorical columns in terms of 9 datasets. In comparison to One-Hot encoding, our method significantly reduces the required encoding space.

Another way for the conversion is by projecting, for example, the work in medGAN [8] has used an Autoencoder to project the input rows to a latent space. However, the performance of data synthesis very much depends on the quality of the latent vectors and the loss of information due to compression can lead to the generation of distorted samples as shown in their experimental results.

The overall data flow of our method is illustrated in Fig 2, to deal with the conversion problem, we propose a

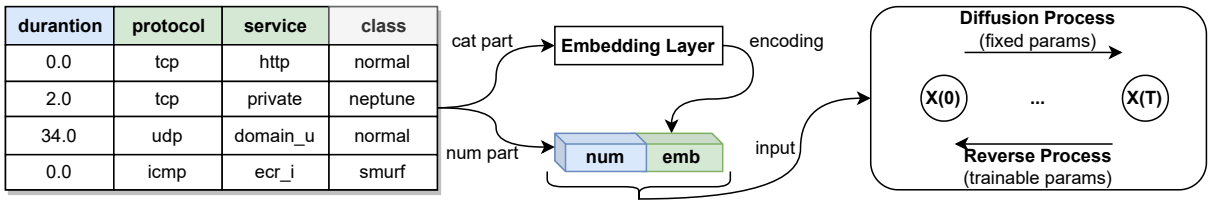

Fig. 2. The overall data flow of our method. On the left is a sample table, where numerical columns are colored in blue and categorical columns are colored in green. Through the trained embedding layer, categories are converted to embeddings. Then, the numerical vectors and embeddings are concatenated as the input for the diffusion model.

method that learns embeddings to represent categories, the categorical columns' dimensions are reduced according to a designed logarithm function, as shown in Fig 1, our method significantly reduces the required encoding space.

Nowadays, privacy protection is becoming more and more important due to people's concerns, and the requirement of regulations, which has led to the formation of data silos. So researchers proposed a new machine learning paradigm to better utilize distributed datasets, called Federated Learning (FL) [9], it is a distributed learning framework that trains a Deep Learning model across multiple decentralized devices holding local data samples, without exchanging them. The data remains locally, only the models travel between the devices and the central server, so the scalability of the model is important.

Besides facing the problem of data silos, the data generated by the model may be at risk of privacy breaches. This is when Differential Privacy (DP) [10] becomes helpful. In general, there are two ways to add noise to meet DP constraints, one is adding noise to the input layer [11] [12], and the other is adding noise to the gradients during training of Deep Learning models [10]. We use the second way to apply DP in our work.

In summary, the proposed end-to-end decentralized tabular synthesizing framework, AutoDiffuse, contains three novelties and contributions. First, we propose a novel method to represent categorical features in tabular data as embeddings. We thus improve the performance of Diffusion Generative Model (DGM) in generating tabular data and facilitates the training of the federated Diffusion Model, which is aligning with Federated Learning training framework, as using compact representations of categorical features leads to a smaller model size to reduce communication burden. To the best of our knowledge, this is the first work that applies FL framework in DGM training, including the embedding and learning tasks. Last, we show the potential of applying DP into this framework to further protect privacy, making the generated data more reliable.

We extensively evaluate AutoDiffuse on 9 common datasets, against 4 state-of-the-art tabular generative models. We consider three criteria, i.e., downstream machine learning utility based on synthetic data, discriminator measure, and coverage of data distributions. Our method outperforms existing central methods by 10.7% and 31.4% in machine learning utility and diversity on average. And it can achieve higher machine learning utility by 6.8% and diversity by 21.1% than another method while having non-IID data on-premise.

2 Background

2.1 Deep Generative Models

TABLE 1
Pros and Cons of Different Models. LL: Log-Likelihood estimation.

GMs	Pros	Cons
VAE	diverse & LL	low sample quality
GAN	fast inference & high quality	no LL & low diversity
AR	LL & high quality	slow
NF	diverse & LL & high quality	slow
DGM	diverse & high quality & LL	slow inference

Variational Autoencoders (VAEs) [13] are latent variable models in which the encoder approximates the posterior distribution of latent variable $P(z|x)$ and the decoder learns to reconstruct the data from posterior samples, they are trained by maximizing a surrogate loss, called Evidence Lower BOund (ELBO). With the precise matching of the latent space, VAE models can generate samples with high diversity. However, according to experimental results, VAE models tend to produce unrealistic, low-quality samples.

Generative Adversarial Networks (GANs) [14] have proven highly successful in the generation of images, and can generate high-quality samples, but their training process is known to be notoriously unstable [15], and the models are susceptible to the mode collapse issue [16]. Since GANs are complete black-box models, there is no estimation for probability density.

Autoregressive Networks (ARs) are generative sequential models and show promise in traditional sequence challenges like natural language generation and audio generation. They factorize joint distribution over a sequence of units $P(x_1, x_2, ..., x_N)$ into conditionals $P(x_k|x_1, x_2, ..., x_{k-1})$. Due to their sequential nature, ARs cannot be trained in parallel leading to slow training and inferring.

Normalizing Flows (NFs) [17] allow exact evaluation of likelihood via change of variables and hence can be directly trained with Maximum Likelihood Estimation (MLE). As NFs require multiple passes through the data to transform it into a standard normal distribution, it can be slow, computationally expensive, and careful design of the structure is necessary for effective performance.

Diffusion Generative Models (DGMs) [18] [5] are a class of probabilistic models that can be used to generate synthetic data. These models are based on the idea of a random process that diffuses through a high-dimensional

space, starting from a fixed point and eventually reaching a stationary distribution. The diffusion process can be thought of as gradually adding calibrated noises which transforms the data distribution into a simple distribution that is easy to sample from, like an isotropic Gaussian distribution, the model that learns to revert this process step by step is utilized as the generator. This approach enables a more straightforward task than the direct generation of data in a single forward pass, as is the case with GANs and VAEs.

[5] provides a general formalism of DGMs, which utilizes differential equations to describe both the corruption process and the inverse process. The corruption process is modeled with the following Stochastic Differential Equation (SDE):

$$dx = f(x,t)dt + g(t)dw \tag{1}$$

where w is the standard Wiener process, f is the *drift* coefficient, g is the *diffusion* coefficient, and time step t ranges from 0 to T. Then the inverse process can be described by the following SDE:

$$dx = \left(f(x,t) - g(t)^2 \bigtriangledown_x logp_t(x)\right) dt + g(t)d\bar{w} \tag{2}$$

where $\bar{w}$ is the standard Wiener process in reversed time, $s(x,t) := \bigtriangledown_x logp_t(x)$ is the so-called *score function*. The essential idea of training a score prediction model is to match the gradients of log probability density functions given x and t [19]:

$$min\left(\left(\hat{s}(x,t) - \bigtriangledown_x logp_t(x)\right)^2\right) \tag{3}$$

The estimate $\hat{s}(x,t)$ can then be plugged into 2 to produce samples in inverse order. And it turns out that the evolution of x over time can be estimated deterministically with an Ordinary Differential Equation (ODE), which can be used to get an unbiased estimation of the true data likelihood:

$$dx = \left(f(x,t) - \frac{1}{2}g(t)^2 \bigtriangledown_x logp_t(x)\right) dt \tag{4}$$

However, the reverse process of a DGM always involves quite an amount of steps, each of which may require separate inference procedures, and may require the use of approximative inference techniques, such as Monte Carlo sampling, which can further increase the computational burden. As a result, despite the high quality and diversity of synthetic samples, DGMs often perform slowly in inference stage.

Table 1 summarizes all the Pros and Cons of the abovementioned deep generative models.

In our experiment section, works using VAE, GAN, and DGM as the structure of tabular generators are chosen as the baseline models.

2.2 Federated Learning

Federated Learning (FL) was first brought up by [9] to tackle the problem of training Deep Learning models without accessing the raw data. Their experiments show that FL can be made practical, as FedAvg trains high-quality models using relatively few rounds of communication, also as demonstrated by results on a variety of model architectures.

The following equation describes how the clients' models are trained with FedAvg:

$$\underset{x\in\mathbb{R}^d}{min}\left[F(x) = \sum_{k=1}^{K}\frac{n_k}{n}F_k(x)\right] \tag{5}$$

$$F_k(x) = \frac{1}{n_k}\sum_{i\in\mathcal{P}_k} f(x;s_i) \tag{6}$$

where K is the number of local models, $n_k = |\mathcal{P}_k|$ is the size of k-th local dataset, s_i is the i-th sample, and $f(x;s_i)$ is the loss function on s_i.

Li et. al. [20] introduce another FL framework, FedProx, to tackle heterogeneity of both data and system in federated networks, however, according to the experimental results, FedAvg still performs better than other carefully adjusted FL algorithms in the scenario where clients' local data are Independent and Identically Distributed (IID), and it can be competitive when clients' data are non-IID.

2.3 Differential Privacy

Differential privacy (DP) is a technique for protecting privacy by adding noise to data while maintaining availability and effectiveness. More precisely, suppose there are two datasets D_1 and D_2, they are said to be neighboring if D_1 can be obtained by adding or removing one record from D_2. Differential Privacy hides the influence of one data sample by constraining the outputs of a mechanism over two neighboring datasets to make it probabilistically indistinguishable. A randomized mechanism $\mathcal{M}$ satisfies (ϵ,δ)-DP [21] if for $\mathcal{S} \subseteq range(\mathcal{M})$ and for all pairs of neighboring datasets D_1 and D_2,

$$Pr\left[\mathcal{M}(D_1) \in \mathcal{S}\right] \leq exp(\epsilon)Pr\left[\mathcal{M}(D_1) \in \mathcal{S}\right] + \delta. \tag{7}$$

When training a neural network under DP constraints, the mechanism called DP-SGD [10] is applied to the set of gradients computed from a minibatch to ensure the privacy of all parameter updates.

2.4 Evaluation Metrics for Tabular Data Synthesizers

Due to the complexity of evaluating generative models, various metrics have been developed to provide a comprehensive assessment from different perspectives.

Machine learning Utility (MLU). The generated dataset should be able to serve as a substitute for the real dataset in the training process. To evaluate the performance of this substitution, metric like Machine Learning Utility (MLU) [22] is used, where prediction models like classifiers or regressors trained on synthetic datasets are tested on real test data and the scores are compared to the performance of the same models when trained on the original real dataset. The performance of the classification tasks is evaluated using accuracy and Macro $F1$, and the regression tasks using R^2. This measure assesses the quality of the synthetic dataset in the context of training useful Machine Learning models.

Discriminator measure (Disc). In order to determine whether the generated data is indistinguishable from the original data, a discriminator, for example, a Random Forest [23] classifier with tuned hyperparameters, is trained on a dataset that consists of a mix of the generated train set (labeled as class 0) and the original train set (labeled as class 1). This measurement is called Discriminator measure (Disc). The model's performance is evaluated on a test set that contains equal proportions of samples from the generated test set and the real test set, and the resulting accuracy is reported.

Coverage. The measurement of diversity can be intuitively determined by the proportion of real samples covered by the synthetic samples. A novel metric called Coverage, proposed by [24], presents an improvement over the other metrics by constructing nearest neighbor manifolds around the real samples, instead of the synthetic samples, as the latter may contain outliers. This approach reduces the computational burden of computing nearest neighbors for each model and focuses on per-dataset manifold computation. Coverage quantifies the fraction of real samples whose neighborhoods contain at least one synthetic sample. It is important to note that Coverage is bounded within the range of 0 to 1.

3 Related Work

3.1 Tabular Data Synthesizers

Tabular data is a ubiquitous presence in fields such as business, finance, science, and technology, and it is playing a central role in data analysis and decision-making. Therefore, designing an efficient tabular data generator is important for synthesizing additional data which can be useful for increasing the size of a dataset, particularly when real-world data is limited, and alleviating privacy concerns as synthesized data preserves the statistical properties of the original data but contains less sensitive information.

Classical statistical learning methods for generating tabular data include Copulas [25] [26], Bayesian Networks [27], Synthetic Minority Oversampling Technique (SMOTE) [28], and Random Over-Sampling (ROS). Statistical tabular data synthesizers, while possessing clear and interpretable structures, also have inherent limitations such as strong assumptions about the shape of the distribution, and risk of privacy breach. These limitations may hinder the data fidelity and practicality of these methods.

Motivated by the fact that Deep Learning methods can automatically model complex, non-linear relationships from the data, many works have tried to solve the tabular data synthesis task with a wide variety of generative deep neural networks, such as Variatioanl Autoencoder (VAE) [22], Generative Adversarial Network (GAN) [22] [29] [8], Autoregressive model (AR) [30], Normalizing Flow based model (NF) [31], and the state-of-the-art generative model, Diffusion Generative Model (DGM) [6], [7]. In general, most of works use One-Hot encoding as the data preprocessing technique to mapping categorical values into numerical vectors consisting of zeroes and ones. Except works like [31] and [8], where the discrete columns are first converted into continuous values by variational dequantization or the input are converted into latent vectors by a trained Autoencoder.

The work closest to ours is [8], in which an Autoencoder is trained to learn a latent space for both categorical and numerical columns of the input data. However, ours mainly uses an Autoencoder as the framework for training embeddings of the categorical features, and the embeddings are used with fixed weights in the following generative process. In this way, only the loosely distributed information of discrete features are compressed, and computational load are reduced.

3.2 Privacy-preserving Designs

Privacy-preserving synthesizers have a number of applications, including the ability to provide synthetic data for training machine learning models, or to allow data analysts to perform statistical analysis without having access to the original data. These synthesizers can be particularly useful in situations where the original data is protected by privacy laws or regulations such as GDPR[32].

Federated learning is a machine learning approach that allows multiple parties to train a model on their datasets without sharing their individual datasets. Previous works like [33] have explored the potential of the federated tabular data synthesizer. However, they do not have many comprehensive indicators and detailed experiments, as well as the risk of privacy leakage in the preprocessing stage.

A differentially private tabular data synthesizer can be used to generate synthetic versions of sensitive tabular data while preserving the privacy of the original data. This is achieved by adding noise to the raw data or training gradients in a controlled manner, such that the resulting data is statistically similar to the original data but does not reveal any sensitive information about individual records. Models such as PATE-GAN [34], and DPDM [35] provide generative models with certain differential privacy guarantees. Our work has explored DP and FL approaches for privacy guarantees.

4 Proposed Method

4.1 Autoencoder as the Entity Embedding Trainer

Categorical features, which may have data types such as String, Integer, and Boolean, are commonly encoded using methods such as One-Hot encoding or Ordinal encoding. These encoders map categorical values to numerical values that are more easily processed by machine learning algorithms. However, these encoding methods have inherent limitations. For instance, One-Hot encoding represents discrete values as binary vectors of length equal to the number of categories, it can suffer from scalability issues when applied to datasets with a large number of categorical columns or categories. On the other hand, Ordinal encoding, which assigns an integer value to each category based on its rank in a pre-defined order, may introduce additional irrelevant information into the input.

To overcome such problems, we propose a framework consisting of categorical features encoder that maps categories into entity embeddings, and the corresponding decoder. The categorical columns' dimensions are reduced

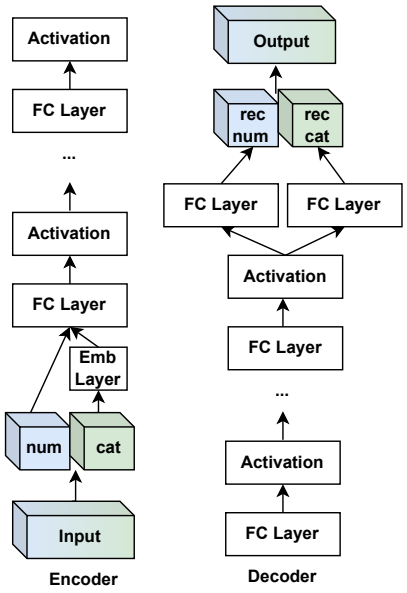

Fig. 3. The architecture of the Autoencoder, consisting of an embedding layer, Fully Connected layers (FCNN), activation layers.

based on the designed logarithm function Eq 8. Entity embedding strategy not only reduces memory usage and speeds up neural networks compared with One-Hot encoding, but more importantly by mapping similar values close to each other in the embedding space it reveals the intrinsic properties of the categorical variables, which helps the neural network to generalize better when the data is sparse and some statistical properties are unclear. The framework is wrapped up as an easy-to-use library shared in [36].

$$f(x) = \begin{cases} 1 & 2 \leq x \\ 2 & 2 < x < 8 \\ 3 & 8 \leq x < 32 \\ 4 & 32 \leq x \end{cases} \tag{8}$$

The unsupervised trainer for learning the embeddings' encoder and decoder is a carefully designed Autoencoder, its overall architecture is shown in Fig 3, which consists of embedding layers, fully connected layers, and activation layers. The layers are designed to incorporate downsampling and upsampling operations for encoding and decoding processes. To further enable the model to learn more information, neurons that process different types of features are trained with different loss functions, such that the categorical part is trained with Cross Entropy loss 9, the numerical part is trained with Mean Square Error (MSE) 10. The central training process is shown in Alg 1.

$$-\sum_{i=1}^{D} y \log(p) \tag{9}$$

$$\sum_{i=1}^{D} (x_i - y_i)^2 \tag{10}$$

Existing work like [37] has also developed ways of mapping categorical features into Euclidean space, however, in their work the mapping is learned by a neural network in its standard supervised training process, which is not suitable when we do not have a specific label. And the embeddings in their work are used for downstream prediction work, unlike our situation where our generation task needs to map generated embeddings back to the original category space.

Algorithm 1 Autoencoder Central Training

Input: tabular dataset X, model θ

1: **for all** each mini-batch x_i of dataset X **do**
2: Split x_i into x_i^{cat} and x_i^{num}
3: Update θ based on Equation 9 and Equation 10
4: **end for**

TABLE 2
Sampling quality in terms of Machine Learning Utility and Discriminator Measure (with NN decoder)

Datasets	Methods	MLU	Disc
Adult	AutoDiffuse(MLP)	0.777±0.006	**0.533±0.015**
	AutoDiffuse(KNN)	0.777±0.006	0.646±0.007
King	AutoDiffuse(MLP)	0.878±0.022	**0.547±0.038**
	AutoDiffuse(KNN)	0.878±0.022	0.553±0.001
Cover	AutoDiffuse(MLP)	0.713±0.008	**0.544±0.01**
	AutoDiffuse(KNN)	0.713±0.008	0.612±0.02

Therefore, the choice of decoder becomes a problem worth thinking about. The first choice is using K Nearest Neighbor (KNN) method decodes the embedding purely based on the distance metrics such as Euclidean distance or Cosine distance, as shown in Table 2, utilizing such method as embeddings decoder can achieve comparable performance on Machine Learning Utility but leads to bad results on Discriminator Measure. This is reasonable since it decides categories only based on the categorical embedding space whereas ignores the correlation between numerical columns.

That's why we choose to use the Multi-Layer Perceptions (MLP) as the decoder instead, which will take all columns into consideration when trying to decode categorical embeddings. The experimental results in Table 2 shows MLP-as-decoder outperforms NN-as-decoder in both MLE and Disc, especially in Disc where MLP-as-decoder makes a large margin.

4.2 Tabular data synthesizer

Since the categorical features are mapped into a continuous space, our method can be simply applied with any type of Score-based Diffusion Model, for example, Gaussian Diffusion Model [18].

The training and inference process is shown in Alg 2 and Alg 3, respectively. The training of a Gaussian Diffusion Model can be simply seen as training a Gaussian noise prediction model like in line 6∼8 Alg 2, and in the inference stage, the predictor will be used to reduce noise from the noisy data as in line 2∼4 Alg 3.

Our proposed entity encoder and decoder are applied as shown in Alg 2 line 4∼5, after the split of raw data x_i, and in Alg 3 line 7∼8, after the split of generated data x_0.

4.3 Federated training strategy

Federated Learning is a private-by-design model training framework, compared with central training which under

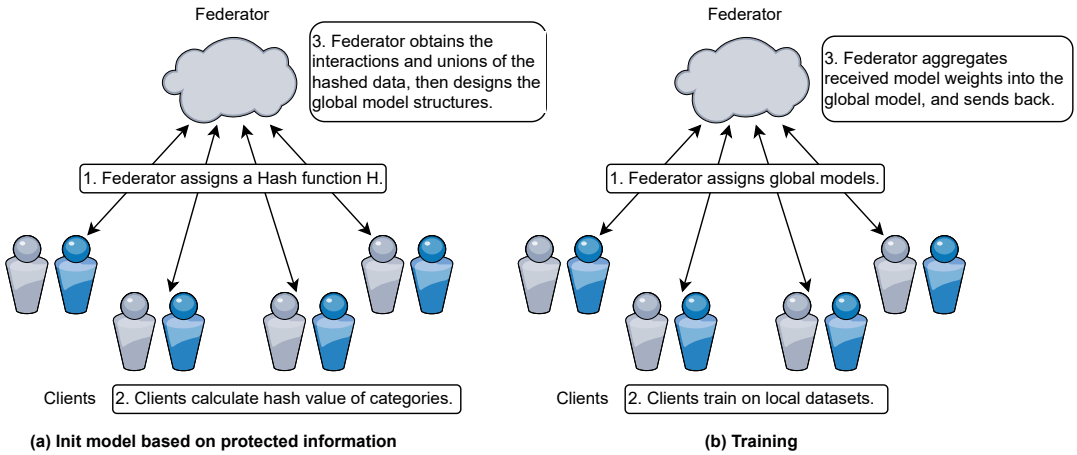

Fig. 4. Overview of the federated training process. Stage 1. (a) Init model based on clients' protected information; Stage 2. (b) Federated training.

Algorithm 2 Tabular Data Synthesizer Training Process

Input: tabular dataset X, embedding lookup table θ_{emb}, model θ_m

for e in epochs **do**
  for each mini-batch x_i of dataset X **do**
    Split x_i into x_i^{cat} and x_i^{num}
    $x_i^{emb} := \theta_{emb}(x_i^{cat})$
    $x_i^{new} := [x_i^{emb}; x_i^{num}]$
    $\epsilon \sim N(0, I)$
    $t \sim Uniform(1, ..., T)$
    Update noise prediction model θ with $\bigtriangledown_\theta = \|\epsilon - \theta_m(x_i^{new}, t)\|^2$
  end for
end for

Algorithm 3 Tabular Data Synthesizer Inference Process

Input: decoder θ_{de}, model θ_m, hyparameters α and σ
Output: generated data x^{gen}

$x_T \sim N(0, 1)$
for $t = T, ..., 1$ **do**
  $z \sim N(0, 1)$ if $t > 1$, else $z = 0$
  $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\theta_m(x_t, t)) + \sigma_t z$ [18]
end for
Split x_0 into x_0^{emb} and x_0^{num}
$x_0^{cat} := \theta_{de}(x_0^{emb}, x_0^{num})$
$x^{gen} := [x_0^{cat}; x_0^{num}]$

the assumption of directly accessible training data, FL can feature decentralized learning on client's local data with a privacy-preserving capability.

The design of a federated tabular data synthesizer is inevitably accompanied by the challenge of designing a simple and effective decentralized data preprocessing framework that ensures privacy protection. Prior work such as [33] requires collecting column statistics like class frequencies of categorical data before synthesizer training, which could be vulnerable to attackers.

In AutoDiffuse, this issue is addressed by replacing the direct collection of statistics with decentralized training of neural networks, which prevents the central server from accessing any client's statistical information. Our federated training framework applies a slightly modified version of the FedAvg algorithm [9], the entity encoder is trained at the beginning with fewer communication rounds, and experimental results demonstrate the feasibility of this setting. The training of diffusion models is also set with proper hyperparameters. The federated training process is shown as Fig 4.

Stage 1. Initialization The federator distributes a Hash function H to all participants, the participants use this Hash function to calculate the hash value of the categories, and send the hashed data to the federator to obtain the intersection and the union. Then the federator uses such information to construct the label encoders and set the number of layers and parameters of the global model structures of the Autoencoder and the Diffusion model, and sends bask to the participants.

Stage 2. Autoencoder training The participants use their own local data to train entity encoders and decoders through the unsupervised training of the Autoencoder. After several local epochs of training, the participants send their model weights to the federator. The federator then aggregates the received model weights according to Eq 5 and Eq 6 as the global model, and sends it back to the participants to start the next round until the preset number of rounds is reached.

Stage 3. Diffusion model training The federator distributes the global entity encoder, decoder, and noise prediction model, the participants then use the global models to train their noise predictor. After training locally for several epochs, the federator collects the local models and aggregates them into the global predictor according to Eq 5 and Eq 6, then sends them back. The participants start the next round of training with the global model until the predetermined number of rounds has been reached.

4.4 Differential privacy

To protect the privacy of individual data points, we apply DP-SGD [10] during the unsupervised training of the entity embeddings, and in the following training of the Diffusion

TABLE 3
Datasets and their descriptions

Name	#Train	#Val	#Test	#Num	#Cat	Task type
Adult	26048	6513	16281	6	8	Binclass
King	13832	3458	4323	17	3	Regression
Cover	25583	6396	7995	10	2	Multiclass
Intrusion	25848	6463	8078	22	16	Multiclass
Insurance	856	214	268	3	3	Regression
Churn	6400	1600	2000	7	4	Binclass
Cardio	44800	11200	14000	5	6	Binclass
Shoppers	7891	1973	2466	10	7	Binclass
Default	19200	4800	6000	15	9	Binclass

model, the input vectors will be partially align with the (ϵ, δ)-DP constrains according to the Post-Processing property of Gaussian Mechanism [38]. In the context of Differential Privacy, the balance between accuracy and privacy preservation is a trade-off. It becomes the responsibility of algorithm designers to make determine regarding the extent to which they are willing to compromise accuracy in order to ensure the protection of sensitive data. As the numerical part are generally normalized before sent to the model, which mitigates the effects of outliers, to some extent, obfuscates some privacy information, and considering the experimental results, our method has opted for partial noise addition.

5 Experiments

We collect 9 common datasets for experiments, the overview list of the datasets is in Table 3. The choice of the datasets follows the standard of having as much categorical columns and categories as possible, since modeling and generating such datasets is a challenge, affecting not only computational resource management but also synthesized data quality.

Our experiments were conducted using a machine that is equipped with an NVIDIA GeForce RTX 2080 Ti GPU.

We choose three tabular data synthesizers as baseline methods covering different types of deep generative models, such as Variational Autoencoder (VAE), Generative Adversarial Network (GAN) and Diffusion Generative Model (DGM). Precisely, the VAE-based baseline is proposed in [22], namely TVAE, it shows effective and efficient data generation ability on most datasets. In [39], the authors propose a GAN-based tabular generator called CTabGAN+ as an enhancement of their previous work CtabGAN [29], the advanced version shows higher data utility and faster convergence, therefore, we only compare our results with CTabGAN+ in our experiments. For the DPM-based baseline, we choose TabDDPM [6], which is the first work to investigate the prospect of the diffusion modelling framework in the field of tabular data.

We choose three metrics for thorough evaluation:

Machine Learning Utility (MLU) The primary evaluation measure is MLU [22]. To elaborate further, machine learning utility quantifies the performance of classification or regression models that are trained on synthetic data and evaluated on the real test set. Intuitively, models trained on high-quality synthetics should be competitive (or even superior) to models trained on real data. In our experiments, we evaluate Machine Learning Utility with the CatBoost model [40] as the prediction model, which is the leading Gradient Boosted Decision Tree implementation providing state-of-the-art performance on tabular tasks.

Discriminator Measure (Disc) Discriminator Measure (Disc) serves the purpose of quantifying the extent of similarity between generated data and the original data. To elaborate further, a discriminator, for example, a Random Forest classifier with carefully optimized hyperparameters, is trained on a composite dataset containing both the generated train set (designated as class 0) and the original train set (designated as class 1). Subsequently, the performance of the model is assessed using a test set that is equally balanced in terms of samples from both the generated test set and the raw test set. The resulting accuracy of the discriminator on this evaluation is reported. And the closer it is to 0.5, the better the generator performs, as the discriminator can not distinguish fake data from real data. Therefore we use Discriminator Measure (Disc) to quantify the real modeling ability of the generator, a good generator should be able to learn the underlying distribution instead of remembering data. Experimental results evaluated on both MLU and Disc is shown and explained in Section 5.1.

Coverage For the quantitative evaluation of the sampling diversity, we use the coverage score [24], which is bounded between 0 and 1. Coverage is the ratio of real records that have at least one fake record in its manifold. A manifold is a sphere around the sample with radius r, where r is the distance between the sample and the k-th nearest neighborhood. And their work is wrapped as an easy-to-use library [41].

5.1 Sampling Quality Analysis

We first discuss the results of Machine Learning Utility and Discriminator Measure before addressing sampling diversity.

A high-quality synthetic dataset should have minimal differences in the MLU trained on real and synthetic data for both classification and regression tasks. Additionally, it should be able to deceive the discriminator into recognizing generated data as original data.

We report macro F1 for classification tasks, R2 for regression tasks. And using the absolute value between Disc and 0.5 to quantify the ability to trick the discriminator into thinking the generated data is real. Table 4 displays the report for all the datasets except Intrusion, which will be discussed separately. AutoDiffuse shows better Machine Learning Utility and Discriminator Measure, which means it outperforms existing methods in terms of sampling quality. Table 10 and Table 11 displays full results on all dataset, they further prove the capability of AutoDiffuse.

Especially on the Intrusion dataset, as shown in Table 10, all the SOTA methods failed due to the difficulty of modeling a large number of categories and categorical columns. Modeling data in a high dimensional space would be a difficulty for DL models, especially ones having a huge number of parameters. TVAE, known for its efficiency and fast convergence, has displayed comparative results on MLU to our method. However, bad performance on Disc as shown in Table 11 indicates TVAE only remembers the

TABLE 4
Summary of experimental results in terms of Machine Learning Utility (MLU) and Discriminator measure (Disc). We report the average of two data quality metrics over all datasets except the dataset Intrusion

Methods	F1 (↑)	R2 (↑)	Diff (↓)
TVAE	0.704	0.701	0.335
CTabGAN+	0.716	0.736	0.21
TabDDPM	0.781	0.831	0.048
AutoDiffuse	**0.803**	**0.865**	**0.035**

TABLE 5
Summary of experimental results in terms of Coverage. We report the average of the data diversity metric over all datasets except the dataset Intrusion

Methods	Coverage
TVAE	0.553
CTabGAN+	0.675
TabDDPM	0.943
AutoDiffuse	**0.951**

data and tries to copy them directly, rather than learning the distribution.

While we were conducting the experiments, work STaSy [7] was published, so we add some experiments with it as another baseline model. The results 6 show that our model can perform better than their work in all metrics.

5.2 Sampling Diversity Analysis

For the quantitative evaluation of the sampling diversity, we use the coverage score [24], which is bounded between 0 and 1. Coverage measures the presence of generated samples within a specified spherical region surrounding real samples. Table 5 summarizes the averaged coverage of each method. TVAE and CTabGAN performs not bad in terms of the sampling quality, whereas it shows rather inferior coverage performance than DGMs. In specific, in the dataset Intrusion, AutoDiffuse has a coverage of 0.763, while all three baselines, TVAE, CTabGAN, TabDDPM, have coverage scores less than 0.1 as in Table 12.

5.3 Applying Differential Privacy

To demonstrate that our method can be used for user privacy protection with Differential Privacy, we apply Gaussian Differential Privacy during the Autoencoder training, and the result shown in Table 8 suggesting when the appropriate parameters are set, the model's performance will not be affected by the addition of noise. A general guideline is to set delta to a value smaller than the inverse of the size of the training dataset, and we set $\delta = 1e-5$, as none of the dataset sizes 3 exceed the reciprocal of this value.

TABLE 6
Evaluation results on more datasets

Datasets	Methods	MLU (F1)	Disc	Coverage
Shoppers	TabDDPM	0.77±0.006	0.54±0.03	0.942±0.005
	Stasy	0.783±0.005	0.927±0.004	0.76±0.008
	AutoDiffuse	**0.797±0.009**	**0.534±0.014**	0.921±0.005
	Real	0.799±0.01	-	-
Default	TabDDPM	0.692±0.003	0.895±0.005	0.14±0.001
	Stasy	0.679±0.013	0.473±0.013	0.092±0.005
	AutoDiffuse	**0.694±0.004**	**0.535±0.0**	**0.295±0.002**
	Real	0.681±0.003	-	-

TABLE 7
Federated training settings of Diffusion Model based tabular synthesizers

Method	Dataset	Clients	Comm rounds	Local epochs	Skew col
AutoDiffuse	Adult	3	10	500	y
TabDDPM	Adult	3	30	1000	y
AutoDiffuse	King	3	20	500	0
TabDDPM	King	3	30	1000	0
AutoDiffuse	Cover	3	30	1000	y
TabDDPM	Cover	3	60	500	y
AutoDiffuse	Intruison	3	40	500	y
TabDDPM	Intruison	3	60	500	y
AutoDiffuse	Insurance	3	5	1000	0
TabDDPM	Insurance	3	10	1000	0
AutoDiffuse	Churn	3	5	1000	y
TabDDPM	Churn	3	30	1000	y
AutoDiffuse	Cardio	3	4	500	y
TabDDPM	Cardio	3	30	1000	y

TABLE 8
Evaluation on sampling quality and diversity over all datasets (with $\delta = 1e-5$)

Datasets	ϵ	MLU (↑)	Diff (↓)	Coverage (↑)
Adult	5.0	0.792±0.004	0.605±0.013	0.841±0.002
King	1.0	0.829±0.003	0.522±0.002	0.940±0.005
Cover	10.0	0.715±0.001	0.516±0.002	0.946±0.001
Intrusion	1.0	0.405±0.013	0.586±0.008	0.64±0.014
Insurance	100.0	0.816±0.004	0.671±0.007	0.916±0.014
Churn	10.0	0.76±0.007	0.547±0.008	0.991±0.001
Cardion	1.0	0.725±0.002	0.508±0.005	0.869±0.001

5.4 Federated Training Results

Recently, Federated Learning (FL) has emerged as a decentralized learning paradigm that enables training on local clients' data while preserving privacy. In the following experimental results, we show our method can be applied to FL framework without losing much effectiveness, settings of the systems trained in the distributed fashion is shown in Table 7.

In FL, the Dirichlet distribution is usually used to create skew motivated by the need to simulate the data heterogeneity across different participants participating in the federated training process [42]. The Dirichlet distribution is a multivariate probability distribution defined on the unit simplex, which is a geometric object that represents all possible probability vectors. By sampling from the Dirichlet distribution, it is possible to generate random probability vectors that can be interpreted as distribution weights or proportions. Specifically, we sample $p_k \sim Dir_N(\alpha)$ and allocate a p_k, j proportion of the instances of class k to party j. Here Dir$(\cdot)$ denotes the Dirichlet distribution and α is the concentration parameter ($\alpha > 0$) used to control the imbalance level of the quantity skew, larger α means larger skew. In the experiments, we use $\alpha = 1.0$ and $\alpha = 10.0$ to test the robustness and generalizability of our design, so that to cover different degrees of heterogeneity. The imbalance

is placed based on the label column if the dataset is for classification, and on a chosen categorical column if the dataset is for regression.

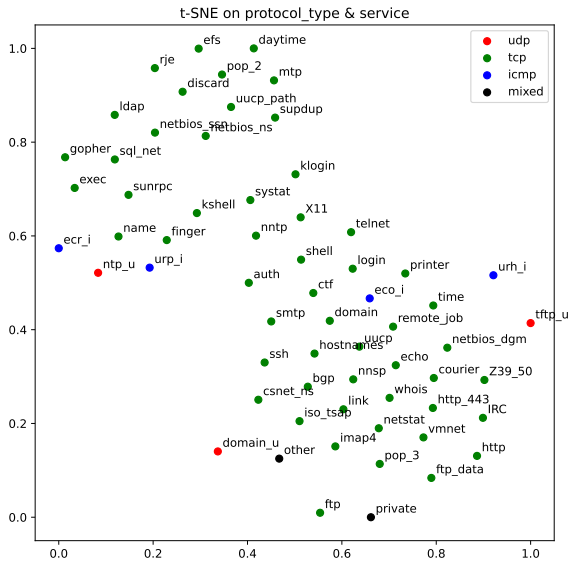

Fig. 5. Centrally trained embeddings visualization via t-SNE dimensionality reduction on columns protocol type and service of Intrusion dataset. The autoencoder for learning entity embeddings is trained centrally. Service = {udp, tcp, icmp, mixed}. Different type of service is represented by 4 different colors, protocol type is labeled on the upper of each dot.

As a FL system has fully distributed participants as well as distributed datasets, training of every part of the central system needs to be generalized into a distributed fashion. Our design of entity encoder can be simply incorporated into the FL framework to construct an end-to-end system by properly setting the hyperparameters. Evaluation of training federated entity encoders is shown in Table 13, on both scenarios of $\alpha = 1.0$ and $\alpha = 10.0$, AutoDiffusewith federated entity encoder could achieve comparable results to AutoDiffusewith centrally trained entity encoder. After preset number of communication rounds, the global model is evaluated on the same test data as the centrally trained model. It is worth noticing that on datasets like Intrusion, Cardio, this is probably because aggregating the distributed entity encoder could lead to more representative embeddings. As shown in the embedding visulization, dimensionality reduction technique t-SNE [43] is applied to the trained embeddings, by comparing centrally trained embeddings in Fig 5 and the embeddings in final rounds of trained by federated learning fashion in Fig 6 and Fig 7, we could see that through aggregating and distributed learning, the location of dots in different colors consist of different clusters with clean separation, which means the embeddings become more expressive. This phenomenon may be due to the fact that federated Autoencoders have more training rounds, as indicated by the settings in Table 13, they generally have 2 to 4 times more training epochs than central ones.

Fig 6, Fig 7 and Table 13 together demonstrate the federated training of an Autoencoder is efficient, commonly

TABLE 9
Sampling quality and diversity averaged over all datasets except Intrusion

α	Methods	MLU (↑)	Diff (↓)	Coverage (↑)
10.0	FL-AutoDiffuse	0.776	0.082	0.954
	FL-TabDDPM	0.734	0.131	0.79
1.0	FL-AutoDiffuse	0.758	0.073	0.946
	FL-TabDDPM	0.71	0.138	0.781

just needs 2 to 4 global communication rounds. The results indicate that the effectiveness of embeddings varies with the number of global epochs. Specifically, less epochs are sufficient for training effective embeddings. The visualization in Fig 7 suggests that increasing the number of epochs does not necessarily make the embeddings more informative, as the separation of the clusters in different colors do not change much at the final rounds such as Round 3, Round 4 and Round 5. Therefore, we choose a small but effective round number for federated Autoencoder training in our experiments.

In addition to training entity encoders and decoders in a distributed manner, our whole system also demonstrates promising results when applied to the FL framework, as in Table 9, showing the averaged experimental results, and altogether with Table 14, Table 15 and Table 16, showing the superior of federated AutoDiffuseover federated TabDDPM in two skew settings.

6 Conclusion

In this paper, motivated by the impressive performance of Diffusion Generative Models, as well as the aim of alleviating nonnegligible privacy concerns, we focus on Diffusion Generative Models for tabular data synthesizing while protecting individual privacy. We identify the limitations of existing tabular data synthesizers and propose a novel encoding strategy for categorical features that improve scalability and modeling performance. Our privacy-preserving end-to-end distributed framework is another novel contribution. We conduct experiments to demonstrate improved performance over existing methods in data quality and diversity, specifically, our proposed approach surpasses SOTA central methodologies by an average of 10.7% and 31.4% in data quality and diversity. Moreover, in situations involving non-IID data, our method exhibits performance gains of 6.8% and 21.1%. The paper concludes by thoroughly evaluating the proposed method through various metrics and showing its superiority over existing baselines.

For future work, evaluating scalability and efficiency for the proposed federated Diffusion Generative Model for tabular data on larger datasets with high-dimensional discrete features and more participants, and incorporating Differential Privacy into the federated training process are possible directions.

References

[1] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

TABLE 10
Machine Learning Utility Evaluation on 7 datasets

Methods	Adult (F1)	King (R2)	Cover (F1)	Intrusion (F1)	Insurance (R2)	Churn (F1)	Cardio (F1)
TVAE	0.777±0.006	0.743±0.027	0.482±0.02	0.373±0.036	0.659±0.116	0.711±0.017	0.678±0.036
CTabGAN+	0.785±0.005	0.735±0.185	0.483±0.058	0.096±0.032	0.737±0.074	0.713±0.022	0.732±0.001
TabDDPM	0.793±0.001	0.858±0.004	0.645±0.014	0.001±0.002	0.803±0.002	0.741±0.006	0.735±0.001
Real	0.811±0.006	0.885±0.011	0.781±0.012	0.763±0.051	0.865±0.015	0.761±0.005	0.728±0.003
AutoDiffuse	0.786±0.005	**0.878±0.022**	**0.713±0.008**	**0.402±0.097**	**0.852±0.014**	**0.763±0.005**	**0.735±0.001**

TABLE 11
Discriminator Measure Evaluation on 7 datasets

Methods	Adult	King	Cover	Intrusion	Insurance	Churn	Cardio
TVAE	0.878±0.019	0.938±0.007	0.67±0.025	1.0±0.0	0.746±0.063	0.846±0.036	0.93±0.009
CTabGAN+	0.623±0.018	0.89±0.042	0.694±0.099	1.0±0.0	0.618±0.091	0.676±0.041	0.758±0.022
TabDDPM	0.568±0.013	0.551±0.009	0.445±0.062	0.999±0.001	0.536±0.028	0.568±0.01	0.487±0.006
AutoDiffuse	**0.533±0.015**	**0.547±0.038**	**0.544±0.01**	**0.555±0.012**	**0.52±0.016**	**0.553±0.005**	**0.49±0.005**

TABLE 12
Coverage Evaluation on 7 datasets

Methods	Adult	King	Cover	Intrusion	Insurance	Churn	Cardio
TVAE	0.408±0.029	0.242±0.008	0.496±0.021	0.004±0.0	0.851±0.069	0.626±0.019	0.081±0.03
CTabGAN+	0.82±0.013	0.307±0.122	0.436±0.189	0.0±0.0	0.857±0.079	0.795±0.086	0.863±0.002
TabDDPM	0.907±0.001	0.903±0.0	0.916±0.003	0.0±0.0	0.998±0.0	0.989±0.0	0.977±0.0
AutoDiffuse	0.907±0.011	**0.989±0.01**	**0.929±0.014**	**0.763±0.003**	0.988±0.004	**0.992±0.001**	0.955±0.001

TABLE 13
AutoDiffuse with federated entity encoders and decoders

Dataset	Clients	Comm rounds	Local epochs	Skew col	Alpha	MLU	DISC	COVERAGE
Adult	3	2	5000	y	10.0	0.785±0.004	0.579±0.002	0.875±0.01
Adult	3	2	5000	y	1.0	0.784±0.008	0.592±0.019	0.864±0.007
Adult (central)	-	-	-	-	-	0.786±0.005	0.533±0.015	0.907±0.011
King	3	2	2000	0	10.0	0.852±0.006	0.548±0.019	0.993±0.001
King	3	2	2000	0	1.0	0.865±0.001	0.552±0.021	0.989±0.006
King (central)	-	-	-	-	-	0.878±0.022	0.547±0.038	0.989±0.01
Cover	3	2	3000	y	10.0	0.716±0.012	0.574±0.0	0.865±0.002
Cover	3	2	3000	y	1.0	0.675±0.03	0.615±0.005	0.797±0.006
Cover (central)	-	-	-	-	-	0.713±0.008	0.544±0.01	0.929±0.014
Intrusion	3	2	500	y	10.0	0.521±0.003	0.54±0.008	0.773±0.002
Intrusion	3	2	500	y	1.0	0.442±0.002	0.556±0.014	0.776±0.012
Intrusion (central)	-	-	-	-	-	0.402±0.097	0.555±0.012	0.763±0.003
Insurance	3	4	3000	0	10.0	0.837±0.007	0.516±0.004	0.989±0.001
Insurance	3	4	3000	0	1.0	0.835±0.005	0.574±0.006	0.978±0.006
Insurance (central)	-	-	-	-	-	0.852±0.014	0.52±0.016	0.988±0.004
Churn	3	3	3000	y	10.0	0.774±0.003	0.581±0.012	0.973±0.005
Churn	3	3	3000	y	1.0	0.774±0.017	0.55±0.008	0.987±0.014
Churn (central)	-	-	-	-	-	0.763±0.005	0.553±0.005	0.992±0.001
Cardio	3	2	2000	y	10.0	0.738±0.012	0.501±0.01	0.954±0.019
Cardio	3	2	2000	y	1.0	0.739±0.013	0.53±0.008	0.959±0.003
Cardio (central)	-	-	-	-	-	0.735±0.001	0.49±0.005	0.955±0.001

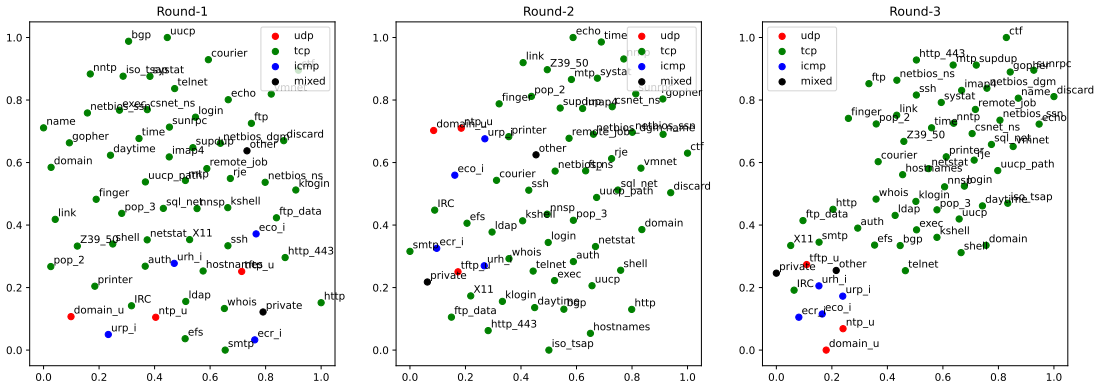

Fig. 6. Embedding visualization on columns protocol type and service of Intrusion dataset

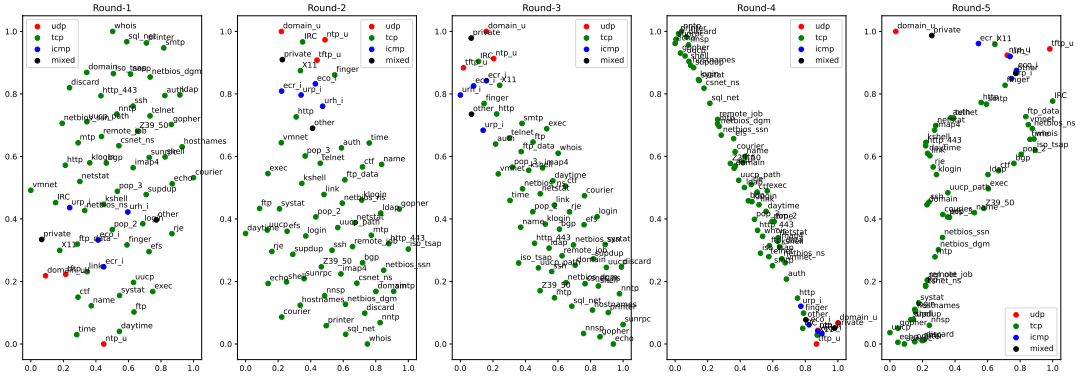

Fig. 7. Embedding visualization on columns protocol type and service of Intrusion dataset.

TABLE 14
Federated training MLU results of Diffusion Model based tabular synthesizers

Alpha	Methods	Adult	King	Cover	Intrusion	Insurance	Churn	Cardio
10.0	AutoDiffuse	**0.783±0.007**	0.851±0.009	**0.691±0.002**	**0.342±0.007**	**0.832±0.006**	**0.762±0.015**	**0.735±0.004**
10.0	TabDDPM	0.669±0.008	0.86±0.017	0.607±0.014	0.0±0.008	0.807±0.007	0.733±0.0	0.727±0.005
1.0	AutoDiffuse	**0.781±0.01**	**0.847±0.004**	**0.652±0.012**	**0.305±0.019**	0.792±0.015	**0.741±0.003**	**0.735±0.006**
1.0	TabDDPM	0.598±0.008	0.844±0.008	0.576±0.004	0.009±0.007	0.796±0.016	0.725±0.006	0.724±0.007

TABLE 15
Federated training DISC results of Diffusion Model based tabular synthesizers

Alpha	Methods	Adult	King	Cover	Intrusion	Insurance	Churn	Cardio
10.0	AutoDiffuse	**0.62±0.0**	0.699±0.013	**0.556±0.002**	**0.513±0.007**	**0.558±0.016**	**0.549±0.004**	**0.512±0.02**
10.0	TabDDPM	0.998±0.003	0.538±0.002	0.431±0.019	1.0±0.005	0.569±0.002	0.578±0.014	0.465±0.02
1.0	AutoDiffuse	**0.634±0.012**	0.62±0.004	**0.511±0.013**	**0.482±0.004**	0.591±0.018	**0.562±0.006**	**0.517±0.015**
1.0	TabDDPM	0.972±0.006	0.582±0.008	0.426±0.006	1.0±0.004	0.541±0.005	0.572±0.01	0.587±0.005

TABLE 16
Federated training coverage results of Diffusion Model based tabular synthesizers

Alpha	Methods	Adult	King	Cover	Intrusion	Insurance	Churn	Cardio
10.0	AutoDiffuse	**0.793±0.013**	**0.967±0.002**	**0.991±0.018**	**0.659±0.01**	0.98±0.01	**0.999±0.007**	0.996±0.007
10.0	TabDDPM	0.007±0.008	0.9±0.004	0.884±0.004	0.0±0.007	0.992±0.001	0.984±0.013	0.975±0.009
1.0	AutoDiffuse	**0.788±0.009**	**0.952±0.01**	**0.997±0.0**	**0.637±0.003**	**0.97±0.004**	**0.996±0.01**	**0.976±0.001**
1.0	TabDDPM	0.002±0.005	0.877±0.001	0.884±0.006	0.0±0.001	0.964±0.007	0.983±0.01	0.975±0.004

[2] "Openml," https://www.openml.org/search?type=data&sort=runs&id=574&status=active.

[3] "Credit card fraud detection — kaggle," https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud.

[4] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2022.

[5] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[6] A. Kotelnikov, D. Baranchuk, I. Rubachev, and A. Babenko, "Tabddpm: Modelling tabular data with diffusion models," *arXiv preprint arXiv:2209.15421*, 2022.

[7] J. Kim, C. Lee, and N. Park, "Stasy: Score-based tabular data synthesis," *arXiv preprint arXiv:2210.04018*, 2022.

[8] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun, "Generating multi-label discrete patient records using generative adversarial networks," in *Machine learning for healthcare conference*. PMLR, 2017, pp. 286–305.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

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

[11] K. Fukuchi, Q. K. Tran, and J. Sakuma, "Differentially private empirical risk minimization with input perturbation," in *Discovery Science: 20th International Conference, DS 2017, Kyoto, Japan, October 15–17, 2017, Proceedings 20*. Springer, 2017, pp. 82–90.

[12] Y. Kang, Y. Liu, B. Niu, X. Tong, L. Zhang, and W. Wang, "Input perturbation: A new paradigm between central and local differential privacy," *arXiv preprint arXiv:2002.08570*, 2020.

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

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[15] C. Chu, K. Minami, and K. Fukumizu, "Smoothness and stability in gans," *arXiv preprint arXiv:2002.04185*, 2020.

[16] H. Thanh-Tung and T. Tran, "Catastrophic forgetting and mode collapse in gans," in *2020 international joint conference on neural networks (ijcnn)*. IEEE, 2020, pp. 1–10.

[17] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International conference on machine learning*. PMLR, 2015, pp. 1530–1538.

[18] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[19] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[20] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, 2018.

[21] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*. Springer, 2006, pp. 265–284.

[22] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[23] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[24] M. F. Naeem, S. J. Oh, Y. Uh, Y. Choi, and J. Yoo, "Reliable fidelity and diversity metrics for generative models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7176–7185.

[25] N. Patki, R. Wedge, and K. Veeramachaneni, "The synthetic data vault," in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2016, pp. 399–410.

[26] Y. Sun, A. Cuesta-Infante, and K. Veeramachaneni, "Learning vine copula models for synthetic data generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 5049–5057.

[27] J. Zhang, G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao, "Privbayes: Private data release via bayesian networks," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 4, pp. 1–41, 2017.

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

[29] Z. Zhao, A. Kunar, R. Birke, and L. Y. Chen, "Ctab-gan: Effective table data synthesizing," in *Asian Conference on Machine Learning*. PMLR, 2021, pp. 97–112.

[30] V. Borisov, K. Seßler, T. Leemann, M. Pawelczyk, and G. Kasneci, "Language models are realistic tabular data generators," *arXiv preprint arXiv:2210.06280*, 2022.

[31] J. Lee, M. Kim, Y. Jeong, and Y. Ro, "Differentially private normalizing flows for synthetic tabular data generation," 2022.

[32] 2018 reform of eu data protection rules. European Commission. [Online]. Available: https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf

[33] Z. Zhao, R. Birke, A. Kunar, and L. Y. Chen, "Fed-tgan: Federated learning framework for synthesizing tabular data," *arXiv preprint arXiv:2108.07927*, 2021.

[34] J. Jordon, J. Yoon, and M. Van Der Schaar, "Pate-gan: Generating synthetic data with differential privacy guarantees," in *International conference on learning representations*, 2018.

[35] T. Dockhorn, T. Cao, A. Vahdat, and K. Kreis, "Differentially private diffusion models," *arXiv preprint arXiv:2210.09929*, 2022.

[36] "entity encoder," https://github.com/xjmxmt/EntityEncoder.

[37] C. Guo and F. Berkhahn, "Entity embeddings of categorical variables," *arXiv preprint arXiv:1604.06737*, 2016.

[38] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[39] Z. Zhao, A. Kunar, R. Birke, and L. Y. Chen, "Ctab-gan+: Enhancing tabular data synthesis," *arXiv preprint arXiv:2204.00401*, 2022.

[40] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *Advances in neural information processing systems*, vol. 31, 2018.

[41] "prdc," https://github.com/clovaai/generative-evaluation-prdc.

[42] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-iid data silos: An experimental study," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 965–978.

[43] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," *Advances in neural information processing systems*, vol. 15, 2002.

2

Conclusion

This project focuses on the synthesis of tabular data using Diffusion Generative Models, driven by their remarkable performance. The study identifies limitations in existing tabular data synthesizers and proposes an innovative encoding strategy for categorical features that improves scalability and modeling performance. Additionally, due to the need to address privacy concerns, a privacy-preserving end-to-end federated framework is introduced.

Experimental evaluations are conducted to assess the proposed method's performance against existing work in terms of data quality and diversity. The results demonstrate notable improvements, surpassing state-of-the-art central methodologies by an average of 10.7% and 31.4% in data quality and diversity, respectively. Furthermore, the proposed method exhibits performance gains of 6.8% and 21.1% in scenarios involving non-IID data in distributed system with multiple clients. The paper concludes by thoroughly evaluating the proposed method through various metrics, establishing its superiority over existing baselines.

In terms of future research directions, evaluating scalability and efficiency for the proposed federated Diffusion Generative Model for tabular data on larger datasets with high-dimensional discrete features and more participants, and trying fulfilling DP constraints during the the federated training process are highlighted. Furthermore, investigating the impact of Different Privacy parameters on synthesized data quality and exploring the application of this method on real-world application scenarios, particularly in domains such as healthcare, finance, and social sciences where privacy concerns are prominent, is also recommended.