CTAB-GAN: Effective Table Data Synthesizing

Zilong Zhao[*] Z.Zhao-8@tudelft.nl
Aditya Kunar[*] A.Kunar@student.tudelft.nl
TU Delft, Delft, The Netherlands.
Robert Birke robert.birke@ch.abb.com
ABB Research Switzerland, Dättwil, Switzerland
Lydia Y. Chen Lydiaychen@ieee.org
TU Delft, Delft, The Netherlands.

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

While data sharing is crucial for knowledge development, privacy concerns and strict regulation (e.g., European General Data Protection Regulation (GDPR)) unfortunately limit its full effectiveness. Synthetic tabular data emerges as an alternative to enable data sharing while fulfilling regulatory and privacy constraints. The state-of-the-art tabular data synthesizers draw methodologies from Generative Adversarial Networks (GAN) and address two main data types in industry, i.e., continuous and categorical. In this paper, we develop CTAB-GAN, a novel conditional table GAN architecture that can effectively model diverse data types, including a mix of continuous and categorical variables. Moreover, we address data imbalance and long tail issues, i.e., certain variables have drastic frequency differences across large values. To achieve those aims, we first introduce the information loss, classification loss and generator loss to the conditional GAN. Secondly, we design a novel conditional vector, which efficiently encodes the mixed data type and skewed distribution of data variable. We extensively evaluate CTAB-GAN with the state of the art GANs that generate synthetic tables, in terms of data similarity and analysis utility. The results on five datasets show that the synthetic data of CTAB-GAN remarkably resembles the real data for all three types of variables and results into higher accuracy for five machine learning algorithms, by up to 17%.

Keywords: GAN; Data synthesis; Tabular data; Imbalanced distribution

1. Introduction

"Data is the new oil" is a quote that goes back to 2006, which is credited to mathematician Clive Humby. It has recently picked up more steam after The Economist[1] published a report titled "The world's most valuable resource is no longer oil, but data". Many companies nowadays discover valuable business insights from various internal and external data sources. However, the big knowledge behind big data often impedes personal privacy and leads to unjustified analysis (Narayanan and Shmatikov (2008)). To prevent the abuse of data and the risks of privacy breaching, the European Commission introduced the European General

[*] Equal contribution

1. https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data

Data Protection Regulation (GDPR) and enforced strict data protection measures. This however instills a new challenge in the data-driven industries to look for new scientific solutions that can empower big discovery while respecting the constraints of data privacy and governmental regulation.

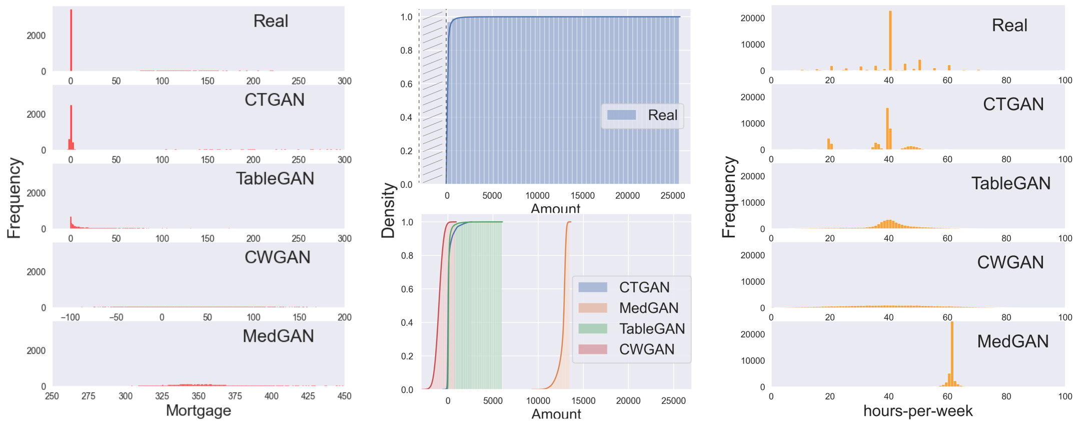
An emerging solution is to leverage synthetic data (Mottini et al. (2018)), which statistically resembles real data and can comply with GDPR due to its synthetic nature. The industrial datasets (at stakeholders like banks, insurance companies, and health care) present multi-fold challenges. First of all, such datasets are organized in tables and populated with both continuous and categorical variables, or a mix of the two, e.g., the value of mortgage for a loan holder. This value can be either 0 (no mortgage) or some continuous positive number. Here, we term such a type of variables, **mixed variable**. Secondly data variables often have a wide range of values as well as skewed frequency distribution, e.g., the statistic of transaction amount for credit card. Most transactions should be within 0 and 500 bucks (i.e. daily shopping for food and clothes), but exceptions of a high transaction amount surely exist.

Generative Adversarial Network (GAN) (Goodfellow et al. (2014)) is one of the emerging data synthesizing methodologies. The GAN is first trained on a real dataset. Then used to generate data. Beyond its success on generating images, Xu et al. (2019); Park et al. (2018); Mottini et al. (2018) have recently applied GAN to generate tabular data. The state of the art of tabular generator (Xu et al. (2019)) treats categorical variables via conditional GAN, where each categorical value is considered as a condition. However, their focus is only on two types of variables, namely continuous and categorical, overlooking an important class of mixed data type. In addition, it is unclear if existing solutions can efficiently handle highly imbalanced categorical variables and skewed continuous variables.

In this paper, we aim to design a tabular data synthesizer that addresses the limitations of the prior state-of-the-art: (i) encoding mixed data type of continuous and categorical variables, (ii) efficient modeling of long tail continuous variables and (iii) increased robustness to imbalanced categorical variables along with skewed continuous variables. Hence, we propose a novel conditional table generative adversarial network, CTAB-GAN. Two key features of CTAB-GAN are the introduction of classification loss in conditional GAN, and novel encoding for the conditional vector that efficiently encodes mixed variables and helps to deal with highly skewed distributions for continuous variables.

We rigorously evaluate CTAB-GAN in three dimensions: (i) utility of machine learning based analysis on the synthetic data, (ii) statistical similarity to the real data, and (iii) privacy preservability. Specifically, the proposed CTAB-GAN is tested on 5 widely used machine learning datasets: Adult, Covertype, Credit, Intrusion and Loan, against 4 state-of-the-art GAN-based tabular data generation algorithms: CTGAN, TableGAN, CWGAN and MedGAN. Our results show that CTAB-GAN not only outperforms all the comparisons in machine learning utility and statistical similarity but also provides better distance-based privacy guarantees than TableGAN, the second best performing algorithm in the machine learning utility and statistical similarly evaluation.

The main contributions of this study can be summarized as follows: (1) Novel conditional adversarial network which introduces a classifier providing additional supervision to improve its utility for ML applications. (2) Efficient modelling of continuous, categorical, and **mixed** variables via novel data encoding and conditional vector. (3) Light-weight data



(a) *Mortgage* in Loan (b) *Amount* in Credit (c) *Hours-per-week* in Adult

Figure 1: Challenges of modeling industrial dataset using existing GAN-based table generator: (a) mixed type, (b) long tail distribution, and (c) skewed data

pre-processing to mitigate the impact of long tail distribution of continuous variables. (4) Providing an effective data synthesizer for the relevant stakeholders.

1.1. Motivation

We empirically demonstrate how the prior state-of-the-art methods fall short in solving challenges in industrial data sets. The detailed experimental setup can be found in Sec. 4.1.

Mixed data type variables. To the best of our knowledge, existing GAN-based tabular generators only consider table columns as either categorical or continuous. However, in reality, a variable can be a mix of these two types, and often variables have missing values. The *Mortgage* variable from the Loan dataset is a good example of mixed variable. Fig. 1(a) shows the distribution of the original and synthetic data generated by 4 state-of-the-art algorithms for this variable. According to the data description, a loan holder can either have no mortgage (0 value) or a mortgage (any positive value). In appearance this variable is not a categorical type due to the numeric nature of the data. So all 4 state-of-the-art algorithms treat this variables as continuous type without capturing the special meaning of the value zero. Hence, all 4 algorithms generate a value around 0 instead of exact 0. And the negative values for Mortgage have no/wrong meaning in the real world.

Long tail distributions. Many real world data can have long tail distributions where most of the occurrences happen near the initial value of the distribution, and rare cases towards the end. Fig. 1(b) plots the cumulative frequency for the original (top) and synthetic (bottom) data generated by 4 state-of-the-art algorithms for the *Amount* in the Credit dataset. This variable represents the transaction amount when using credit cards. One can imagine that most transactions have small amounts, ranging from few bucks to thousands of dollars. However, there definitely exists a very small number of transactions with large amounts. Note that for ease of comparison both plots use the same x-axis, but Real has no negative values. Real data clearly has 99% of occurrences happening at the start of the range, but the distribution extends until around 25000. In comparison none of the synthetic data generators is able to learn and imitate this behavior.

Skewed multi-mode continuous variables. The term *multi-mode* is extended from Variational Gaussian Mixtures (VGM). More details are given in Sec. 3.3. The intuition behind using multiple modes can be easily captured from Fig. 1(c). The figure plots in each row the distribution of the working *Hours-per-week* variable from the Adult dataset. This is not a typical Gaussian distribution. There is an obvious peak at 40 hours but with several other lower peaks, e.g. at 50, 20 and 45. Also the number of people working 20 hours per week is higher than those working 10 or 30 hours per week. This behavior is difficult to capture for the state-of-the-art data generators (see subsequent rows in Fig.1(c)). The closest results are obtained by CTGAN which uses Gaussian mixture estimation for continuous variables. However, CTGAN loses some modes compared to the original distribution.

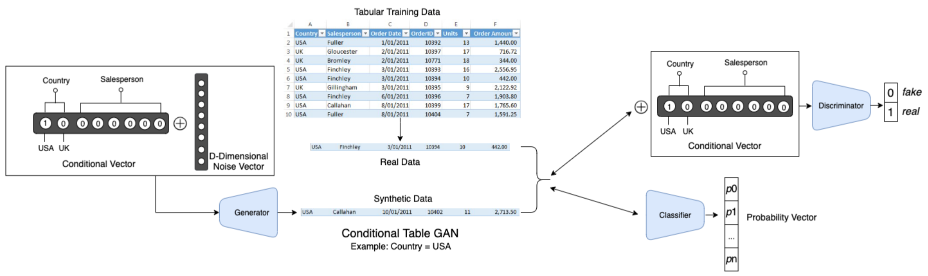
The above examples show the shortcomings of current state-of-the-art GAN-based tabular data generation algorithms and motivate the design of our proposed CTAB-GAN.

2. Related Studies

We divide the related studies using GAN to generate tabular data into two categories: (i) based on GAN, and (ii) based on conditional GAN.

GAN-based generator. Several studies extend GAN to accommodate categorical variables by augmenting GAN architecture. MedGAN (Choi et al. (2017)) combines an auto-encoder with a GAN. It can generate continuous or discrete variables, and has been applied to generate synthetic electronic health record (EHR) data. CrGAN-Cnet (Mottini et al. (2018)) uses GAN to conduct Airline Passenger Name Record Generation. It integrates the Cramér Distance (Bellemare et al. (2017)) and Cross-Net architecture (Wang et al. (2017)) into the algorithm. In addition to generating with continuous and categorical data types, CrGAN-Cnet can also handle missing value in the table by adding new variables. TableGAN (Park et al. (2018)) introduces information loss and a classifier into GAN framework. It specifically adopts Convolutional Neural Network (CNN) for generator, discriminator and classifier. Although aforementioned algorithms can generate tabular data, they cannot specify how to generate from a specific class for particular variable. For example, it is not possible to generate health record for users whose sex is female. In addition to data generation, privacy is another important factor for synthetic tabular data. PATE-GAN (Yoon et al. (2019)) is not specifically designed for tabular data generation, but it proposes a framework which generates synthetic data with differential privacy guarantees.

Conditional GAN-based generator. Due to the limitation of controlling generated data via GAN, Conditional GAN is increasingly used, and its conditional vector can be used to specify to generate a particular class of data. This feature is important when our available data is limited and highly skewed, and we need synthetic data of a specific class to re-balance the distribution. For instance, for preparing starting dataset of online learning scenarios (Zhao et al. (2019, 2021); Younesian et al. (2020)). CW-GAN (Engelmann and Lessmann (2020)) applies the Wasserstein distance (Arjovsky et al. (2017)) into the conditional GAN framework. It leverages the usage of conditional vector to oversample the minority class to address imbalanced tabular data generation. CTGAN (Xu et al. (2019)) integrates PacGAN (Lin et al. (2020)) structure in its discriminator and uses Generator loss and WGAN loss plus gradient penalty (Gulrajani et al. (2017)) to train a conditional GAN framework. It also adopts a strategy called training-by-sampling, which takes advantage of conditional vector,



	A	B	C	D	E	F
1	Country	Salesperson	Order Date	OrderID	Units	Order Amoun
2	USA	Fuller	1/01/2011	10392	13	1,440.00
3	UK	Gloucester	2/01/2011	10397	17	716.72
4	UK	Bromley	2/01/2011	10771	18	344.00
5	USA	Finchley	3/01/2011	10393	16	2,556.95
6	USA	Finchley	3/01/2011	10394	10	442.00
7	UK	Gillingham	3/01/2011	10395	9	2,122.92
8	USA	Finchley	6/01/2011	10396	7	1,903.80
9	USA	Callahan	8/01/2011	10399	17	1,765.60
10	USA	Fuller	8/01/2011	10404	7	1,591.25

USA	Finchley	3/01/2011	10394	10	442.00

USA	Callahan	10/01/2011	10402	11	2,713.50

Figure 2: Synthetic Tabular Data Generation via CTAB-GAN

to deal with the imbalanced categorical variable problem. In our paper, we not only focus on modelling continuous or categorical variables, but also cover the mixed data type (i.e., variables that contain both categorical and continuous values, or even missing values). We effectively combine the strengths of prior art, such as classifier, information, generator loss, effective encoding, and conditional vector. Furthermore, we proactively address the pain point of long tail variable distributions and propose a new conditional vector structure to better deal with imbalanced datasets.

3. CTAB-GAN

CTAB-GAN is a tabular data generator designed to overcome the challenges outlined in Sec. 1.1. In CTAB-GAN we invent a *Mixed-type Encoder* which can better represent mixed categorical-continuous variables as well as missing values. CTAB-GAN is based on a conditional GAN (CGAN) to efficiently treat minority classes, with the addition of classification, information and generator loss (Park et al. (2018); Odena et al. (2017); Xu et al. (2019)) to improve semantic integrity and training stability, respectively. Finally, we leverage a log-frequency sampler to overcome the mode collapse problem for imbalanced variables.

3.1. Technical Background

GANs are a popular method to generate synthetic data first applied with great success to images (Karras et al. (2019)) and later adapted to tabular data (Yahi et al. (2017)). GANs leverage an adversarial game between a generator trying to synthesize realistic data and a discriminator trying to discern synthetic from real samples.

To address the problem of dataset imbalance, we leverage *conditional generator* and *training-by-sampling* methods from CTGAN. The idea behind this is to use an additional vector, termed as the conditional vector, to represent the classes of categorical variables. This vector is both fed to the generator and used to bound the sampling of the real training data to subsets satisfying the condition. We can leverage the condition to resample all classes giving higher chances to minority classes to train the model.

To enhance the generation quality, we incorporate three extra terms in the loss function of the generator: information (Park et al. (2018)), classification (Odena et al. (2017)) and generator loss (Xu et al. (2019)). The information loss penalizes the discrepancy between statistics of the generated data and the real data. This helps to generate data which is statistically closer to the real one. The classification loss requires to add to the GAN architecture an auxiliary classifier in parallel to the discriminator. For each synthesized label the classifier outputs a predicted label. The classification loss quantifies the discrepancy between the synthesized and predicted class. This helps to increase the semantic integrity of synthetic records. For instance, (sex=female, disease=prostate cancer) is not a semantically correct record as women do not have a prostate, and no such record should appear in the original data and is hence not learnt by the classifier. The generator loss measures the difference between the given conditions and the output classes of the generator. This loss helps the generator to learn to produce the exact same classes as the given conditions. Classification loss is used by TableGAN but not CTGAN, since CTGAN does not contain classifier. Generator loss is implemented by CTGAN but not TableGAN, because TableGAN is not a conditional GAN.

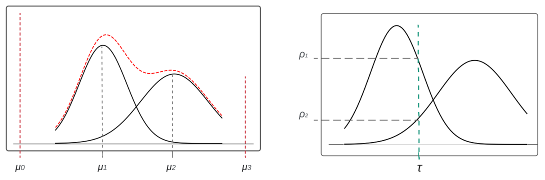
To counter complex distributions in continuous variables we embrace the *mode-specific normalization (MSN)* idea (Xu et al. (2019)) which encodes each value as a value-mode pair stemming from Gaussian mixture model.

3.2. Design of CTAB-GAN

The structure of CTAB-GAN comprises three blocks: Generator $\mathcal{G}$, Discriminator $\mathcal{D}$ and an auxiliary Classifier $\mathcal{C}$ (see Fig. 2). Since our algorithm is based on conditional GAN, the generator requires a noise vector plus a conditional vector. Details on the conditional vector are given in Sec. 3.4. To simplify the figure, we omit the encoding and decoding of the synthetic and real data detailed in Sec. 3.3.

GANs are trained via a zero-sum minimax game where the discriminator tries to maximize the objective, while the generator tries to minimize it. The game can be seen as a mentor ($\mathcal{D}$) providing feedback to a student ($\mathcal{G}$) on the quality of his work. Here, we introduce additional feedback for $\mathcal{G}$ based on the information loss, classification loss and generator loss. The information loss matches the first-order (i.e., mean) and second-order (i.e., standard deviation) statistics of synthesized and real records. This leads the synthetic records to have the same statistical characteristics as the real records. The classification loss equates the correlation between classes and the other variable values. This helps to check the semantic integrity, and penalizes synthesized records where the combination of values are semantically incorrect. Finally, the generator loss is the cross-entropy between the given conditional vector and the generated output classes. It enforces the conditional generator to produce the same classes as the given conditional vector. These three losses are added to the original loss term of $\mathcal{G}$ during training. $\mathcal{G}$ and $\mathcal{D}$ are implemented using CNNs with the same structure as in Park et al. (2018). CNNs are good at capturing the relation between pixels within an image, which in our case, can help to increase the semantic integrity of synthetic data. To process row records stored as vectors with CNN, we wrap the row data into the closest square matrix dimensions, i.e. $d \times d$ where d is the ceiled square root of the row data dimensionality and pad missing values with zeros. $\mathcal{C}$ uses a multi-layer-perceptron (MLP) with four 256-neuron hidden layers. The classifier is trained on the original data to better interpret the semantic integrity. Hence synthetic data are reverse transformed from their matrix encoding to vector (details in Sec. 3.3). Real data is encoded (details in Sec. 3.3 and 3.5) before being used as input for $\mathcal{C}$ to create the class label predictions.

Let f_x and $f_{\mathcal{G}(z)}$ denote the features fed into the softmax layer of $\mathcal{D}$ for a real sample x and a sample generated from latent value z, respectively. The **information loss** for $\mathcal{G}$ is



(*a*) Mixed type (*b*) Mode selection

Figure 3: Encoding for mix data type variable

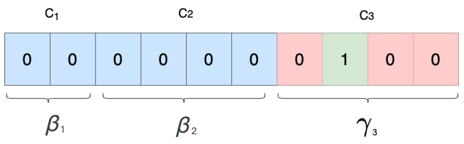


Figure 4: Conditional vector $\mathcal{V}$ example

expressed as $\mathcal{L}^G_{info} = ||\mathbb{E}[f_x]_{x\sim p_{data}(x)} - \mathbb{E}[f_{\mathcal{G}(z)}]_{z\sim p(z)}||_2 + ||\mathbb{SD}[f_x]_{x\sim p_{data}(x)} - \mathbb{SD}[f_{\mathcal{G}(z)}]_{z\sim p(z)}||_2$ where $p_{data}(x)$ and $p(z)$ denote prior distributions for real data and latent variable, $\mathbb{E}$ and $\mathbb{SD}$ denote the mean and standard deviations of the features, respectively. The **classifier loss** is given by $\mathcal{L}^G_{class} = \mathbb{E}[|l(G(z)) - \mathcal{C}(fe(G(z)))|]_{z\sim p(z)}$ where $l(.)$ returns the target label and $fe(.)$ returns the input features of a given row x. Finally, the **generator loss** is given by $\mathcal{L}^G_{generator} = H(m_i, \hat{m}_i)$ where m_i and $\hat{m}_i$ are the given and generated conditional vector bits corresponding to column i and $H(.)$ is the cross-entropy loss. Columns are selected using the training-by-sampling procedure (see Sec. 3.4 for details).

Let $\mathcal{L}^D_{orig}$ and $\mathcal{L}^G_{orig}$ denote the original GAN loss functions from Goodfellow et al. (2014) to train the discriminator $\mathcal{D}$ and generator $\mathcal{G}$, respectively. For $\mathcal{G}$ the complete training objective is the combination of $\mathcal{L}^G = \mathcal{L}^G_{orig} + \mathcal{L}^G_{info} + \mathcal{L}^G_{class} + \mathcal{L}^G_{generator}$, while for $\mathcal{D}$ it is unchanged, i.e. $\mathcal{L}^D_{orig}$. Finally, the loss to train the classifier $\mathcal{C}$ is similar to the classification loss of the generator, i.e. $\mathcal{L}^C_{class} = \mathbb{E}[|l(x) - \mathcal{C}(fe(x))|]_{x\sim p_{data}(x)}$.

3.3. Mixed-type Encoder

The tabular data is encoded variable by variable. We distinguish three types of variables: categorical, continuous and mixed. We define variables as mixed if they contain both categorical and continuous values or continuous values with missing values. We propose the new Mixed-type Encoder to deal with such variables. With this encoder, values of mixed variables are seen as concatenated value-mode pairs. We illustrate the encoding via the exemplary distribution of a mixed variable shown in red in Fig. 3(a). One can see that values can either be exactly μ_0 or μ_3 (the categorical part) or distributed around two peaks in μ_1 and μ_2 (the continuous part). We treat the continuous part using a variational Gaussian mixture model (VGM) (Bishop (2006)) to estimate the number of modes k, e.g. $k = 2$ in our example, and fit a Gaussian mixture. The learned Gaussian mixture is $\mathbb{P} = \sum_{k=1}^{2} \omega_k \mathcal{N}(\mu_k, \sigma_k)$, where $\mathcal{N}$ is the normal distribution and ω_k, μ_k and σ_k are the weight, mean and standard deviation of each mode, respectively.

To encode values in the continuous region of the variable distribution, we associate and normalize each value with the mode having the highest probability (see Fig. 3(b)). Given ρ_1 and ρ_2 being the probability density from the two modes in correspondence of the variable value τ to encode, we select the mode with the highest probability. In our example ρ_1 is higher and we use mode 1 to normalize τ. The normalized value α is: $\alpha = \frac{\tau - \mu_1}{4\sigma_1}$. Moreover we keep track of the mode β used to encode τ via one-hot encoding, e.g. $\beta = [0, 1, 0, 0]$ in our example. The final encoding is giving by the concatenation of α and β: $\alpha \bigoplus \beta$ where $\bigoplus$ is the vector concatenation operator. The categorical part (e.g., μ_0 or μ_3 in Fig. 3(a)) is treated similarly, except α is directly set to 0. Because the category is determined only by one-hot encoding part. For example, for a value in μ_3, the final encoding is given by $0 \bigoplus [0, 0, 0, 1]$. Categorical variables use the same encoding as the continuous intervals of mixed variables. Categorical variables are encoded via a one-hot vector γ. Missing values are treated as a separate unique class and we add an extra bit to the one-hot vector for it. A row with $[1, \ldots, N]$ variables is encoded by concatenation of the encoding of all variable values, i.e. either $(\alpha \bigoplus \beta)$ for continuous and mixed variables or γ for categorical variables. Having n continuous/mixed variables and m categorical variables ($n + m = N$) the final encoding is:

$$\bigoplus_{i=1}^{n} \alpha_i \bigoplus \beta_i \bigoplus_{j=n+1}^{N} \gamma_j \tag{1}$$

3.4. Counter Imbalanced Training Datasets

In CTAB-GAN, we use conditional GAN to counter imbalanced training datasets using training-by-sampling (Xu et al. (2019)), but extended to include the modes of continuous and mixed columns. When we sample real data, we use the conditional vector to filter and rebalance the training data. The conditional vector $\mathcal{V}$ is a bit vector given by the concatenation of all mode one-hot encodings β (for continuous and mixed variables) and all class one-hot encodings γ (for categorical variables) for all variables present in Eq. (1). Each conditional vector specifies a single mode or a class. More in detail, $\mathcal{V}$ is a zero vector with a single one in correspondence to the selected variable with selected mode/class. Fig. 4 shows an example with three variables, one continuous (C_1), one mixed (C_2) and one categorical (C_3), with class 2 selected on C_3.

To rebalance the dataset, each time we need a conditional vector during training, we first randomly choose a variable with uniform probability. Then we calculate the probability distribution of each mode (or class for categorical variables) in that variable using frequency as proxy and sample a mode based on the logarithm of its probability. Using the log probability instead of the original frequency gives minority modes/classes higher chances to appear during training. This helps to alleviate the collapse issue for rare modes/classes. Extending the conditional vector to include the continuous and mixed variables helps to deal with imbalance in the frequency of modes used to represent them. Moreover, since generator is conditioned on all data-types during training, this enhances the learned correlation between all variables.

3.5. Treat Long Tails

We encode continuous values using variational Gaussian mixtures to treat multi-mode data distributions (details in Sec. 3.3). However, Gaussian mixtures can not deal with all types of data distribution, notably distributions with long tail where few rare points are far from the bulk of the data. VGM has difficulty to encode the values towards the tail. To counter this issue we pre-process variables with long tail distributions with a logarithm transformation. For such a variable having values with lower bound l, we replace each value τ with compressed τ^c:

$$\tau^c = \left\{ \begin{array}{cl} \log(\tau) & \text{if } l > 0 \\ \log(\tau - l + \epsilon) & \text{if } l \leqslant 0, \text{ where } \epsilon > 0 \end{array} \right\} \tag{2}$$

Table 1: Description of Datasets.

Dataset	Train/Test Split	Target variable	Continuous	Binary	Multi-class	Mixed-type	Long-tail
Adult	39k/9k	'income'	3	2	7	2	0
Covertype	45k/5k	'Cover_Type'	10	44	1	0	0
Credit	40k/10k	'Class'	30	1	0	0	1
Intrusion	45k/5k	'Class'	22	6	14	0	2
Loan	4k/1k	'PersonalLoan'	5	5	2	1	0

The log-transform allows to compress and reduce the distance between the tail and bulk data making it easier for VGM to encode all values, including tail ones. We show the effectiveness of this simple yet performant method in Sec. 4.6.

4. Experimental Analysis

To show the efficacy of the proposed CTAB-GAN, we select five commonly used machine learning datasets, and compare with four state-of-the-art GAN based tabular data generators. We evaluate the effectiveness of CTAB-GAN in terms of the resulting ML utility, statistical similarity to the real data, and privacy distance. Moreover, we provide an ablation analysis to highlight the efficacy of the unique components of CTAB-GAN.

4.1. Experimental Setup

Datasets. Our algorithm is tested on five commonly used machine learning datasets. Three of them **Adult**, **Covertype** and **Intrusion**are from the UCI machine learning repository[2]. The other two **Credit** and **Loan** are from Kaggle[3]. All five tabular datasets have a target variable, for which we use the rest of the variables to perform classification. Due to computing resource limitations, 50K rows of data are sampled randomly in a stratified manner with respect to the target variable for Covertype, Credit and Intrusion datasets. However, the Adult and Loan datasets are not sampled. The details of each dataset are shown in Tab. 1. One thing to notice is that we assume that the user already knows the data type of each variable before training. Xu et al. (2019) holds the same assumptions.

Baselines. Our CTAB-GAN is compared with 4 state-of-the-art GAN-based tabular data generators: CTGAN, TableGAN, CWGAN and MedGAN. To have a fair comparison, all algorithms are coded using Pytorch, with the generator and discriminator structures matching the descriptions provided in their respective papers. For Gaussian mixture estimation of continuous variables, we use the same settings as the evaluation of CTGAN, i.e. 10 modes. All algorithms are trained for 150 epochs for Adult, Covertype, Credit and Intrusion datasets, whereas the algorithms are trained for 300 epochs on Loan dataset. The reason is Loan dataset is significantly smaller than the others containing only 5000 rows and requires a long training time to converge. Lastly, each experiment is repeated 3 times.

Environment. Experiments are run under Ubuntu 20.04 on a machine equipped with 32 GB memory, a GeForce RTX 2080 Ti GPU and a 10-core Intel i9 CPU.

2. http://archive.ics.uci.edu/ml/datasets
3. https://www.kaggle.com/{mlg-ulb/creditcardfraud,itsmesunil/bank-loan-modelling}

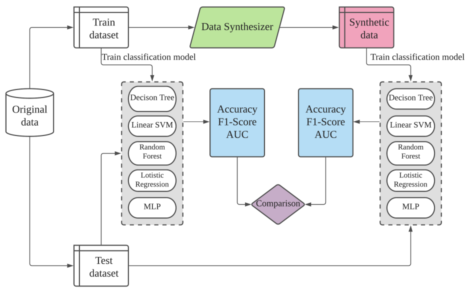


Figure 5: Evaluation for ML utility

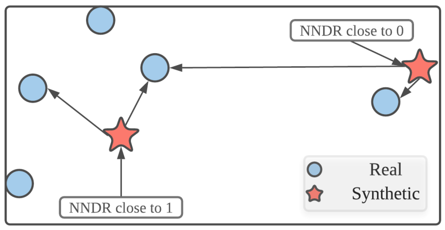


Figure 6: Illustration of NNDR

4.2. Evaluation Metrics

The evaluation is conducted on three dimensions: (1) machine learning (ML) utility, (2) statistical similarity and (3) privacy preservability. The first two measure if the synthetic data can be used as a good proxy of the original data. The third criterion evaluates the nearest neighbour distances within and between the original and synthetic datasets, respectively.

4.2.1. Machine Learning Utility

To quantify the ML utility, we compare the performance achieved by 5 widely used machine learning algorithms on real versus synthetic data: decision tree classifier, linear support-vector-machine (SVM), random forest classifier, multinomial logistic regression and MLP. We use scikit-learn 0.24.2 with default parameters except max-depth 28 for decision tree and random forest, and neurons 128 for MLP.For a fair compassion, all hyper-parameters are fixed across all datasets. Due to this our results can differ slightly from Xu et al. (2019) where the authors use different ML models and hyper-parameters for each dataset. Fig. 5 shows the evaluation process. The train dataset and synthetic dataset are the same size. The aim of this design is to test how close the ML utility is when we train a ML model using the synthetic data vs the real data.

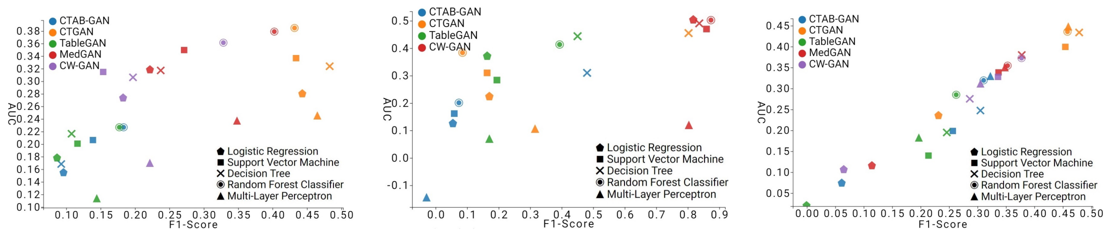
4.2.2. Statistical Similarity

Three metrics are used to quantify the statistical similarity between real and synthetic data.

Jensen-Shannon divergence (JSD). The JSD provides a measure to quantify the difference between the probability mass distributions of individual categorical variables belonging to the real and synthetic datasets, respectively. Moreover, this metric is bounded between 0 and 1 and is symmetric allowing for an easy interpretation of results.

Wasserstein distance (WD). In similar vein, the Wasserstein distance is used to capture how well the distributions of individual continuous/mixed variables are emulated by synthetically produced datasets in correspondence to real datasets. We use WD because we found that the JSD metric was numerically unstable for evaluating the quality of continuous variables, especially when there is no overlap between the synthetic and original dataset. Hence, we resorted to utilize the more stable Wasserstein distance.

Difference in pair-wise correlation (Diff. Corr.). To evaluate how well feature interactions are preserved in the synthetic datasets, we first compute the pair-wise correlation matrix for the columns within real and synthetic datasets individually. Pearson correlation



(a) Covertype (b) Intrusion (c) Loan

Figure 7: ML utilities difference (i.e., AUC and F1-scoree) for five algorithms and five datasets

coefficient is used between any two continuous variables. It ranges between $[-1, +1]$. Similarly, the Theil uncertainty coefficient is used to measure the correlation between any two categorical features. It ranges between $[0, 1]$. And the correlation ratio between categorical and continuous variables is used. It also ranges between $[0, 1]$. Note that the dython[4] library is used to compute these metrics. Finally, the difference between pair-wise correlation matrices for real and synthetic datasets is computed.

4.2.3. Privacy preservability

To quantify the privacy preservability, we resort to distance metrics (instead of differential privacy (Yoon et al. (2019))) as they are intuitive and easy to understand by data science practitioners. Specifically, we use the following metrics to evaluate the privacy risk.

Distance to Closest Record (DCR). The DCR is used to measure the Euclidean distance between any synthetic record and its closest corresponding real neighbour. Ideally, the higher the DCR the lesser the risk of privacy breach. Furthermore, the 5^{th} percentile of this metric is computed to provide a robust estimate of the privacy risk.

Nearest Neighbour Distance Ratio (NNDR). NNDR measures the ratio between the Euclidean distance for the closest and second closest real neighbour to any corresponding synthetic record. NNDR is a commonly used concept in computer vision area for matching local image descriptors. Platzer[5] introduces this idea to evaluate the privacy for synthetic data. This ratio is within $[0, 1]$. Higher values indicate better privacy. Low NNDR values between synthetic and real data may reveal sensitive information from the closest real data record. Fig. 6 illustrates the case. Note that the 5^{th} percentile is computed here as well.

4.3. Results Analysis

ML Utility. Tab. 2 shows the average ML utility difference between real and synthetic data in terms of accuracy, F1 score, and AUC. A better synthetic data is expected to have small differences. It can be seen that CTAB-GAN outperforms all other state-of-the-art methods in terms of Accuracy, F1-score and AUC. Accuracy is the most commonly used classification metric, but since we have imbalanced target variable, F1-score and AUC are more stable metrics for such cases. AUC ranges from 0 to 1. CTAB-GAN largely shortens the AUC difference from 0.169 (best in state-of-the-art) to 0.094.

To obtain a better understanding, Fig. 7 plots the (F1-score, AUC) for all 5 ML models for the Covertype, Intrusion and Loan datasets. Due to the page limit restrictions, results

4. http://shakedzy.xyz/dython/modules/nominal/#compute_associations
5. https://github.com/mostly-ai/virtualdatalab

Table 2: Results against State of the Art on ML Utility, Data Similarity, and Privacy.

Method	**ML Utility Difference**			**Statistical Similarity**			**Privacy Preservation**					
							DCR			**NNDR**		
	Accuracy	**F1-score**	**AUC**	**Avg JSD**	**Avg WD**	**Diff. Corr.**	**R&S**	**R**	**S**	**R&S**	**R**	**S**
CTAB-GAN	**8.90%**	**0.107**	**0.094**	**0.062**	**1197**	**2.09**	1.118	0.428	0.937	0.713	0.414	0.591
CTGAN	21.51%	0.274	0.253	0.0704	1769	2.73	1.517	0.428	1.026	0.763	0.414	0.624
TableGAN	11.40%	0.130	0.169	0.0796	2117	2.30	0.988	0.428	0.920	0.681	0.414	0.632
MedGAN	14.11%	0.282	0.285	0.2135	46257	5.48	1.918	0.428	0.254	0.871	0.414	0.393
CW-GAN	20.06%	0.354	0.299	0.1318	238155	5.82	2.197	0.428	1.124	0.847	0.414	0.675

for Adult and Credit datasets are not shown. Their results are similar to the ones for Covertype (see Fig. 7(a)). Fig. 7(b) shows that for the Intrusion dataset CTAB-GAN largely outperforms all others across all ML models used for evaluation. For datasets such as Covertype, the results of CTAB-GAN and TableGAN are similar and clearly better than the rest. This is because apart from CTGAN, the other models fail to deal with the imbalanced categorical variables. Furthermore, as CTGAN uses a VGM model with 10 modes, it fails to converge to a suitable optimum for Covertype that mostly comprises single mode Gaussian distributions.

For the Loan dataset (see Fig. 7(c)), TableGAN is better than CTAB-GAN and others, but the difference between the two is smaller than for the Intrusion dataset. We believe that the reason CTAB-GAN outperforms the others by such a wide margin (17% higher than second best for averaged accuracy across the 5 machine learning algorithms) for the Intrusion dataset is that it contains many highly imbalanced categorical variables. In addition, Intrusion also includes 3 long tail continuous variables. Our results indicate that none of the state-of-the-art techniques can perform well under these conditions. The Loan dataset is significantly smaller than the other four and has the least number of variables. Moreover, all continuous variables are either simple one mode Gaussian distributions or just uniform distributions. Therefore, we find that the encoding method in CTAB-GAN which works well for complex cases, fails to converge to a better optimum for simple and small datasets.

Statistical similarity. Statistical similarity results are reported in Tab. 2. CTAB-GAN stands out again across all comparisons. For categorical variables (i.e. average JSD), CTAB-GAN outperforms CTGAN and TableGAN by 13.5% and 28.4%. This is due to the use of the conditional vector, the log-frequency sampling and extra losses, which works well for both balanced and imbalanced distributions. For continuous variables (i.e. average WD), we still benefit from the design of the conditional vector. The average WD column shows some extreme numbers such as 46257 and 238155 comparing to 1197 of CTAB-GAN. The reason is that these algorithms generate extremely large values for long tail variables. Besides divergence and distance, our synthetic data also maintains better correlation. We can see that TableGAN also performs well here. However, as the extended conditional vector enhances the training procedure, this helps to maintain even more the correlation between variables. Because the extended conditional vector allows the generator to produce samples conditioned even on a given VGM mode for continuous variables. This increases the capacity to learn the conditional distribution for continuous variables and hence leads to an improvement in the overall feature interactions captured by the model.

Privacy preservability. We use distance-based algorithms to give an overview on privacy in our evaluation. On the one hand, if the distance between real and synthetic data is too large, it simply means that the quality of generated data is poor. On the other

Table 3: Ablation and Training Time Analysis.

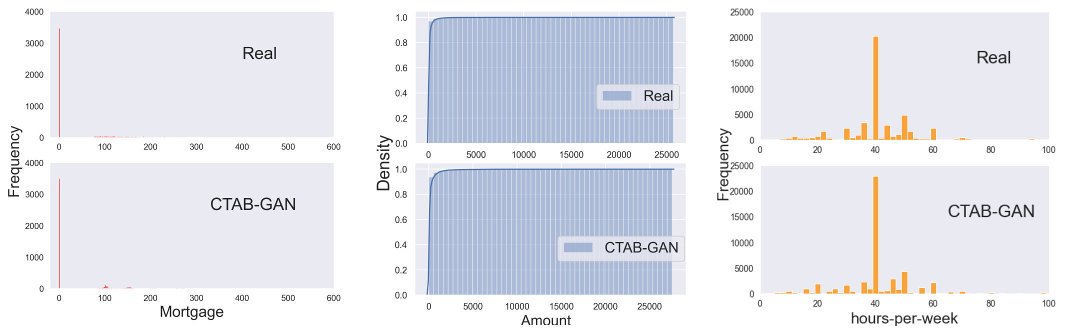
Dataset	Ablation [F1 score]					Training Time [s/epoch]				
	CTAB-GAN	**w/o $\mathcal{C}$**	**w/o I. Loss**	**w/o MSN**	**w/o LT**	**CTAB-GAN**	**CTGAN**	**TableGAN**	**MedGAN**	**CWGAN**
Adult	0.704	-0.01	-0.037	-0.05	-	7.50	1.66	0.52	0.33	4.73
Covertype	0.532	-0.018	-0.184	-0.118	-	8.38	3.32	1.69	0.33	10.41
Credit	0.710	+0.011	-0.177	+0.06	+0.001	8.92	2.42	1.73	0.34	4.50
Intrusion	0.842	-0.031	-0.437	+0.003	-0.074	10.73	2.80	1.75	0.34	10.81
Loan	0.803	-0.044	+0.028	+0.013	-	0.57	0.17	0.06	0.04	1.12

hand, if the distance between real and synthetic data is too small, it means that there is a risk to reveal sensitive information from the training data. Therefore, the evaluation of privacy is relative. The privacy results are shown in Tab. 2. It can be seen that the DCR and NNDR between real and synthetic data (i.e., R&S) all indicate that generation from TableGAN has the shortest distance to real data (i.e., highest privacy risk). In that case, CTAB-GAN not only outperforms TableGAN in ML utility and statistic similarity, but also in all privacy preservability metrics by 11.6% and 4.5% for DCR and NNDR, respectively. Another insight from this table is that for MedGAN, DCR within synthetic data is 41% smaller than within real data. This suggests that it suffers from the mode collapse problem.

4.4. Ablation Analysis

To illustrate the efficiency of each strategy we implement an ablation study which cuts off the different components of CTAB-GAN one by one: (1) **w/o $\mathcal{C}$**. In this experiment, Classifier $\mathcal{C}$ and the corresponding classification loss for Generator $\mathcal{G}$ are taken away from CTAB-GAN; (2) **w/o I. loss** (information loss). In this experiment, we remove information loss from CTAB-GAN; (3) **w/o MSN**. In this case, we substitute the mode specific normalization based on VGM for continuous variables with min-max normalization and use simple one-hot encoding for categorical variables. Here the conditional vector is the same as for CTGAN; (4) **w/o LT** (long tail). In this experiment, long tail treatment is no longer applied. This only affects datasets with long tailed columns, i.e. Credit and Intrusion.

The results are compared with the baseline implementing all strategies. All experiments are repeated 3 times, and results are evaluated on the same 5 machine learning algorithms introduced in Sec. 4.2.1. The test datasets and evaluation flow are the same as shown in Sec. 4.1 and Sec. 4.2. Tab. 3 shows the results. Each part of CTAB-GAN has different impacts on different datasets. For instance, **w/o $\mathcal{C}$** has a negative impact for all datasets except Credit. Since Credit has only 30 continuous variables and one target variable, the semantic check can not be very effective. **w/o information loss** has a positive impact for Loan, but results degenerate for all other datasets. It can even make the model unusable, e.g. for Intrusion. **w/o MSN** performs bad for Covertype, but has little impact for Intrusion. Credit w/o MSN performs better than original CTAB-GAN. This is because out of 30 continuous variables, 28 are nearly single mode Gaussian distributed. The initialized high number of modes, i.e. 10, for each continuous variable (same setting as in CTGAN) degrades the estimation quality. **w/o LT** has the biggest impact on Intrusion, since it contains 2 long tail columns which are important predictors for the target column. For Credit, the influence is limited. Even if the long tail treatment fits well the *amount* column (see Sec. 4.6), this variable is not a strong predictor for the target column. In general, if we average the column values, all the ablation tests have a negative impact for the performance which justifies our design choices for CTAB-GAN.



(a) *Mortgage* in Loan (b) *Amount* in Credit (c) *Hours-per-week* in Adult

Figure 8: Challenges of modeling industrial dataset using existing GAN-based table generator: (a) mixed type, (b) long tail distribution, and (c) skewed data

4.5. Training Time Analysis

The above results show that CTAB-GAN outperforms the State of the art in ML utility, statistical similarity and privacy preservation. This comes at the expense of additional training complexity, i.e., information, classifier and generator losses, and feature encoding, i.e., mixted-type, long tail, and VGM mode in conditional vectors. Tab. 3 shows the training time per epoch for all models on all datasets. As expected, CTAB-GAN is slower than other algorithms. To improve the training efficiency of CTAB-GAN, we may do the following: (1) pre-train the classifier with the real dataset before-hand instead as in parallel to the generator. By storing once the inference results we can speed up the calculation of the classifier loss. (2) use min-max normalization instead of a VGM estimation for continuous columns with a clear single mode Gaussian distribution. Since VGM estimates $k >> 1$ modes, using min-max normalization can significantly reduce the length of the continuous column encoding; (3) treat categorical columns with a clear single mode Gaussian distribution as a continuous column and use min-max normalization instead of one-hot encoding. To convert values back to categorical form, rounding has been shown to work well in Park et al. (2018). (1) is easy to adopt. However, (2) and (3) require human intuition to verify and determine the distribution of each column before training. From our preliminary study based on Adult dataset, CTAB-GAN can reduce training time by 29% implementing (1) and by 12% using (2) and (3) while maintaining a similar quality of generation.

4.6. Results for Motivation Cases

After reviewing all the metrics, let us recall the three motivation cases from Sec. 1.1.

Mixed data type variables. Fig. 8(a) compares the real and CTAB-GAN generated variable *Mortgage* in Loan dataset. CTAB-GAN encodes this variable as mixed type. We can see that CTAB-GAN generates clear 0 values and the frequency is close to real data.

Long tail distributions. Fig. 8(b) compares the cumulative frequency graph for the *Amount* variable in Credit. This variable is a typical long tail distribution. One can see that CTAB-GAN perfectly recovers the real distribution. Due to log-transform data pre-processsing, CTAB-GAN learns this structure significantly better than the state-of-the-art methods shown in Fig. 1(b).

Skewed multi-mode continuous variables. Fig. 8(c) compares the frequency distribution for the continuous variable *Hours-per-week* from Adult. Except the dominant peak at 40, there are many side peaks. Fig. 1(c), shows that TableGAN, CWGAN and MedGAN struggle since they can learn only a simple Gaussian distribution due to the lack of any special treatment for continuous variables. CTGAN, which also use VGM, can detect other modes. Still, CTGAN is not as good as CTAB-GAN. The reason is that CTGAN lacks the mode of continuous variables in the conditional vector. By incorporating the mode of continuous variables into conditional vector, we can apply the training-by-sample and logarithm frequency also to modes. This gives the mode with less weight more chance to appear in the training and avoids the mode collapse.

5. Conclusion

Motivated by the importance of data sharing and fulfillment of governmental regulations, we propose CTAB-GAN– a conditional GAN based tabular data generator. CTAB-GAN advances beyond the prior state-of-the-art methods by modeling mixed variables and provides strong generation capability for imbalanced categorical variables, and continuous variables with complex distributions. To such ends, the core features of CTAB-GAN include (i) introduction of the classifier into conditional GAN, (ii) effective data encoding for mixed variable, and (iii) a novel construction of conditional vectors. We exhaustively evaluate CTAB-GAN against four tabular data generators on a wide range of metrics, namely ML utilities, statistical similarity and privacy preservation. The results show that the synthetic data of CTAB-GAN results into high utilities, high similarity and reasonable privacy guarantee, compared to existing state-of-the-art techniques. The improvement on complex datasets is up to 17% in accuracy comparing to all state-of-the-art algorithms. The remarkable results of CTAB-GAN demonstrate its potential for a wide range of applications that greatly benefit from data sharing, such as banking, insurance, and manufacturing.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th ICML - Volume 70*, page 214–223. JMLR.org, 2017.

Marc G. Bellemare, Ivo Danihelka, W. Dabney, S. Mohamed, Balaji Lakshminarayanan, S. Hoyer, and R. Munos. The cramer distance as a solution to biased wasserstein gradients. *ArXiv*, abs/1705.10743, 2017.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. Generating multi-label discrete patient records using generative adversarial networks. *arXiv preprint arXiv:1703.06490*, 2017.

Justin Engelmann and Stefan Lessmann. Conditional wasserstein gan-based oversampling of tabular data for imbalanced learning. *arXiv preprint arXiv:2008.09202*, 2020.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th NIPS - Volume 2*, page 2672–2680, Cambridge, MA, USA, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *the 31st NIPS*, page 5769–5779, 2017.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE/CVF CVPR*, pages 4396–4405, 2019.

Zinan Lin, Ashish Khetan, Giulia Fanti, and Sewoong Oh. Pacgan: The power of two samples in generative adversarial networks. *IEEE JSAIT*, 1(1):324–335, 2020.

Alejandro Mottini, Alix Lheritier, and Rodrigo Acuna-Agost. Airline Passenger Name Record Generation using Generative Adversarial Networks. In *workshop on Theoretical Foundations and Applications of Deep Generative Models. ICML*, July 2018.

Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, pages 111–125, 2008.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *The 34th ICML - Volume 70*, page 2642–2651, 2017.

Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proc. VLDB Endow.*, 11(10):1071–1083, June 2018.

Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, New York, NY, USA, 2017.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. In *NIPS*, 2019.

Alexandre Yahi, Rami Vanguri, and Noémie Elhadad. Generative adversarial networks for electronic health records: A framework for exploring and evaluating methods for predicting drug-induced laboratory test trajectories. In *NIPS workshop*, 2017.

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *ICLR*, 2019.

Taraneh Younesian, Zilong Zhao, Amirmasoud Ghiassi, Robert Birke, and Lydia Y Chen. Qactor: On-line active learning for noisy labeled stream data. *arXiv preprint arXiv:2001.10399*, 2020.

Zilong Zhao, Sophie Cerf, Robert Birke, Bogdan Robu, Sara Bouchenak, Sonia Ben Mokhtar, and Lydia Y. Chen. Robust anomaly detection on unreliable data. In *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2019.

Zilong Zhao, Robert Birke, Rui Han, Bogdan Robu, Sara Bouchenak, Sonia Ben Mokhtar, and Lydia Y. Chen. Enhancing robustness of on-line learning models on highly noisy data. *IEEE Transactions on Dependable and Secure Computing*, 18(5):2177–2192, 2021.