Extending CTAB-GAN with StackGAN

Orhan Rauf Akdemir
Supervisor(s): Zilong Zhao, Lydia Chen
EEMCS, Delft University of Technology, The Netherlands

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

As privacy regulations (e.g. European General Data Protection Regulation) often prevent valuable flows of data between stakeholders, data synthesis can play a crucial role in sharing captured value in data sets without sharing personal details. Different attempts have been made at solving this problem with Generative Adversarial Networks (GAN) architectures. The aim of this study to create an architecture that can create data that is more resemblant of an original dataset than what the state-of-the-art can achieve, while functioning under the same privacy budget. We apply the concept of progressive learning on the state of the art tabular GAN, i.e., stacking two CTAB-GAN in a sequential manner. Two CTAB-GAN architectures were used, where Stage I CTAB-GAN is trained as usual and the Stage II CTAB-GAN is trained using the result of Stage I CTAB-GAN and the same conditional vector that was used for Stage I. Stage II CTAB-GAN can rectify defects made in Stage I and further refine the the results to finally achieve a generative model that can better capture the information in the original dataset. The challenge lies in creating the condition in which the second GAN can actually improve on the first GANs results. The results on four datasets show that better data synthesis can be achieved with a second GAN that has a fully connected generator, as opposed to a copy of the original generator of CTAB-GAN.

1 Introduction

Data, and intelligence derived from it, has become an integral part of decision-making in many organizations around the world. However, due to privacy regulations there is a limitation on how data can flow between and within organizations. A solution that can be leveraged to bridge this problem, is the synthesis of data. There has been ongoing research to finding a method that can generate data with statistical resemblance of the original data sets while still complying with certain privacy budgets.

Data is often stored in a tabular format that contains both categorical and continuous variables. However, these data points can also follow a mixed-type distribution, where the value can be 0 (for no data) or a continuous value for the case in which there is data.[1] Furthermore, as Zhao et al. (2021) states, "...data variables often have a wide range of values as well as skewed frequency distribution". These properties among other things, are part of the challenge in designing a model that can generate data that is resemblant of the original dataset.

In the last years, GAN have shown impressive results in various fields such as the generation of photos [2], paint art [3] and three-dimensional objects [4]. Despite the relative success of existing techniques for these fields, similar adversarial methods have proven significantly less effective for the tables that contain categorical, continuous and mixed data types. Multiple architectures to solve this problem have been proposed, such as TableGAN [5], CT-GAN [6] and more recently, CTAB-GAN [1]. CTAB-GAN has shown performance that is the state-of-the-art (SOTA) at the time of writing this paper, as measured by Machine Learning (ML) Utility and Statistical Similarity Difference (SSD) [1]. The research done by Zhao et al. and its CTAB-GAN attempt to solve the aforementioned problems of data skew and mixed-type variables, which in turn have an effect on total accuracy of the architecture.

This research described in this paper is a continuation on the work by Zhao et al. More specifically, this research is an attempt to improve on CTAB-GAN by extending it in a fashion that is conceptually similar to the StackGAN [7] architecture. CTAB-GAN has shown good results in the synthesis of data that has a high skew with mixed-type variables. However, in order to be able to make good synthesis of mixed-type variables, the columns need to be encoded *mode-specific Normalization (MSN)* and a one-hot approach. This in turn causes the column space to become large, which has a negative effect on synthesis quality.

As stacking GANs has shown remarkable results in other fields than just image processing [8], the underlying hypothesis for this research is that it can be used to synthesize data in "high resolution" in general. The dimensionality is a challenging factor for the generation of tabular data, and it is this where StackGAN can be useful.

2 Research Questions

The aim of this research is to test if a stacked CTAB-GAB architecture can outperform a single CTAB-GAN architecture for tabular data synthesis, as measured by ML Utility and Statistical Similarity Difference (SSD). The outcomes of the project will contain a codebase that has the stacked CTAB-GAN architecture, a series of benchmark tests that are analagous to and comparable with the CTAB-GAN paper such that the hypothesis of this research can be tested and finally the comparison itself. The main question of this research is:

"Can the CTAB-GAN+ performance, as measured by ML Utility and SSD, be improved by stacking multiple networks under varying privacy budgets?"

The following sub-questions need to be answered in order to answer the main questions:

1. What is the effect of the new architecture on the ML utility
2. What is the effect on the new architecture on SSD?
3. What is the effect on the new architecture on privacy preservability?

3 Related work

We will primarily look at conditional GAN-based generators that have been proposed over the last years. Conditional GAN offer an edge that regular GAN do not have, namely the ability to generate data of a particular class. This is particularly useful for data that is highly skewed, as it allows the trainig process to leverage *training-by-sampling*, as proposed by Engelmann and Stefan Lessman in the CW-GAN paper[9] and uses the Wasserstein distance. The idea is to relatively oversample underrepresented classes in the original dataset, so as to make sure the neural network is

trained for all class types. CT-GAN, which also attempts to synthesize tabular data, employs mode-specific normalization (MSN), which is useful for the modeling of complex data distributions [6]. It encodes each value as a value-mode pair that is gained from a Gaussian mixture model. CTAB-GAN uses all of these methods to enhance its performance[1] and Stacked CTAB-GAN does too.

StackGAN [7] utilizes the conditional vector in a specialized manner. It generates images based on a text-embedding that describes the image that needs to be generated. It is a "stack" of GAN; it has a Stage I GAN and a Stage II GAN. The first GAN is used to generate a low quality image, and the second GAN is used to further refine and correct defects in the output of the first GAN. In tabular data synthesis there are no true images that are generated but instead tabular rows, but analogous logic can be applied to tabular data as well. This same concept is used by Mendéz-Luca et al. for generating molecules from gene expressions[8]. Gene expressions are used as conditional vector and used for the first GAN as well as the second GAN. This method does not employ the same benefit of being able generate a small image first and an upscaling second, but does leverage the ability to inject the conditional vector twice to make a refinement. In this light, Stacked CTAB-GAN is more similar to the architecture proposed by Mendéz-Luca et al. However, their research does not solve the problems proposed in this research, which is the problem of synthesizing tabular data.

As stacking GANs has shown remarkable results in other fields than just image processing [8], the underlying hypothesis for this research is that it can be used to synthesize data in "high resolution" in general. The dimensionality is a challenging factor for the generation of tabular data, and it is this where StackGAN can be useful.

4 Motivation

Although the original StackGAN paper was introduced for text-to-image synthesis [7], StackGAN-inspired architectures show promising results in domains outside of image generation, such as molecule structure generation [8]. As stacked GANs have proven to be useful synthesizers of data in large vector spaces in various context [7; **?**], the assumption was that there is a probability that it can improve the performance of the tabular synthesis of CTAB-GAN as well.

The problem with synthesizing images of high-resolution with GAN is that the model distributions and image distributions are difficult to align with many pixels. The strategy used for StackGAN is first generating a low resolution image with one GAN and later applying a second GAN on the generated image to scale it up to the higher resolution. An intuitive analogy may be a painter who first paints the outline of their painting before working out the details.

Similarly, CTAB-GAN attempts to generate a relatively high-resolution image (which is then transformed to the representation of a row in a table) with a single-layered neural architecture. Due to one-hot encoding of categorical columns and encoding of mixed-type columns, the dimensionality of a table grow larger than its original, which in turn causes more difficulty in the alignment of the model distribution and the real data distribution. By chaining two CTAB-GAN, this problem is partially solved, as the first layer makes an imperfect approximation, after which the the final layer is able to "finish it off". Figure 1 shows the new architecture of the stacked CTAB-GAN.

5 Stacked CTAB-GAN

Stacked CTAB-GAN is a tabular data generator that uses the CTAB-GAN architecture for multiple layers. This includes its Mixed-type Encoder, which as stated in Zhoa et al. ,"can better represent mixed categorical-continuous variables as well as missing values", the calculated selecting of "minority classes" in the generated conditional vectors and the combination of classification, information and generator loss. [1] The fundamental difference is that the new method includes two CTAB-GANs instead of one. The two GANs are trained in the same loop and chained together through the output of the Stage I GAN and input of the Stage II GAN.

5.1 Background on CTAB-GAN

To solve the problem of imbalanced datasets, i.e. the datasets in which certain classes occur more often than others, a conditional generator is used to feed both the generator and discriminator in the training process. [6] This is the aforementioned *training-by-sampling* strategy.

In addition to this, CTAB-GAN uses an auxiliary classifier to be included in the loss function. This classifier is trained to tell the difference between the predicted and synthesized class through cross-correlations [1]. This helps minimize the discrepancy between the generated records and real records as it is trained.

Finally we use the same mode-specific normalization that is used in the CTAB-GAN to handle complicated distributions. [6] This helps capture complex data distributions better. In particular, this is used to treat long tails (data with a with range and high degree of skew) in the data.

5.2 New architecture

The architecture that is proposed with this research is a "stacked" CTAB-GAN; a two-step process in which a first CTAB-GAN is trained and later used to feed the second CTAB-GAN with a synthesized record instead of a noise vector. See Figure 1 for a visual overview of the architecture.

The Stage I GAN is a conditional GAN that is equivalent to CTAB-GAN. CTAB-GAN uses convolutional methods to capture inter-columnar relationships. It transforms a single row to a square image that has pixels greater or equal to the number of columns in the data that is one-hot encoded and MSN-encoded. It is padded with zeroes to fill the square image. The convolutions allow it to capture the relationships between distributions.

The Stage II GAN is another CTAB-GAN that is in many ways similar to the original CTAB-GAN, except for two main differences. Instead of using the concatenation of a

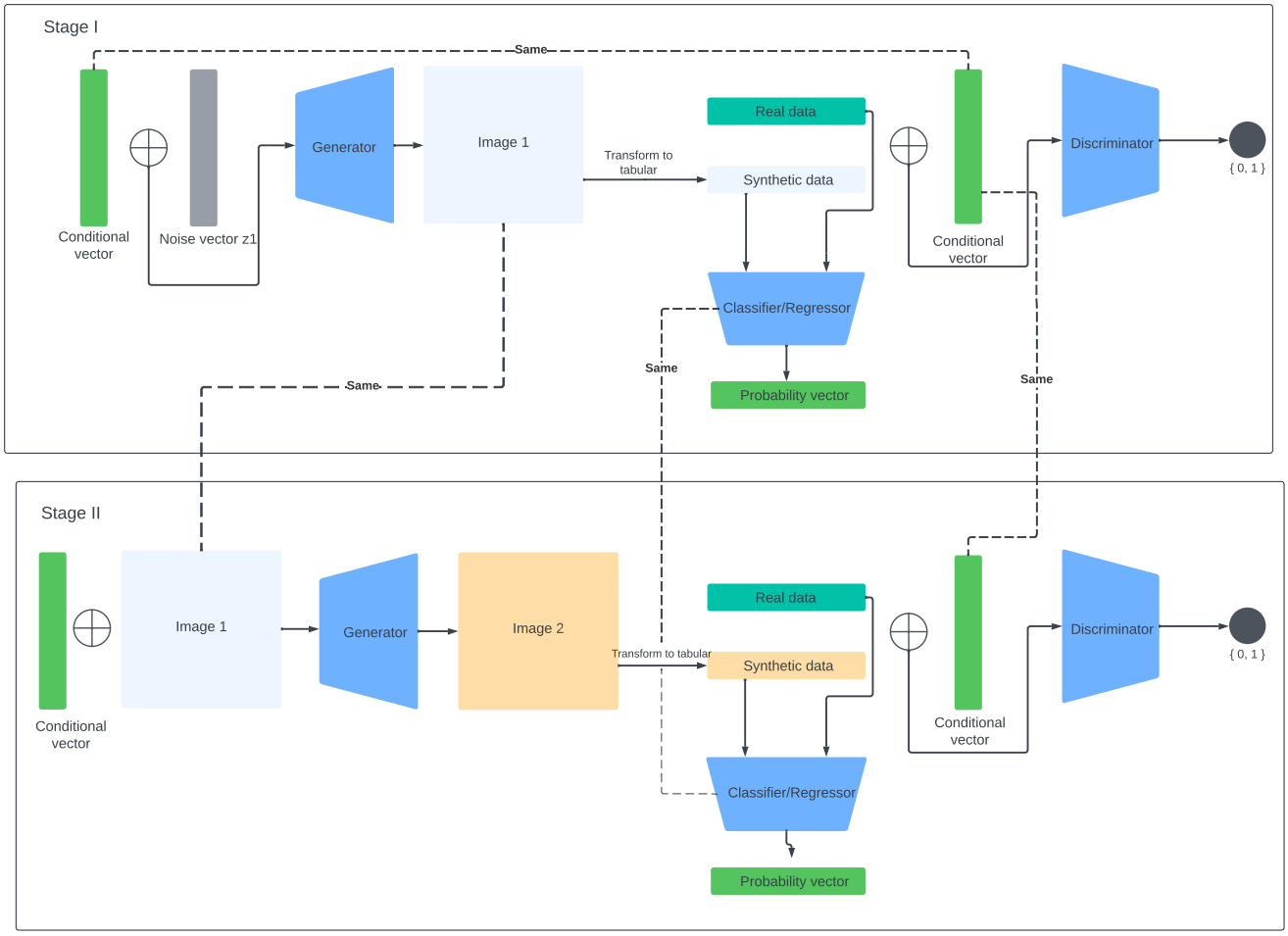

Figure 1: Stacked CTAB-GAN Architecture: it can be seen that there are now two CTAB-GAN chainged together, where the conditional vector is shared among the networks and the image of the first CTAB-GAN is fed into the next layer for training and later, sampling. The classifier and discriminator for the second layer have the same function, but they are separately trained as well.

noise vector from a normal distribution and the generated conditional vector, it takes in a concatenation of the generated record and the same conditional vector that was used to generate that record in the first CTAB-GAN.

5.3 Loss calculation for generator

Both GAN use the same loss functions that are described in Zhao et al. for CTAB-GAN. The loss function contains of multiple parts, that all play a separate role in finally shaping the weights of the generator in the way that is desired. The loss functions that make up the larger function of the generator are the following:

1. Orignal GAN Loss. This loss is the loss described by Goodfellow et al. (2014) and is given by the output of the discriminator. This can be seen as the ”regular” GAN loss that most implementations of GAN inherently have.

2. Information Loss. The information loss is used as a general steering mechanism to make sure the data that is generated is generally resemblant of the original dataset. This is done through two statistical quantifiers: the means and the standard deviations. The higher the difference, the higher the information loss.

3. Generator Loss. CTAB-GAN is a conditional GAN, meaning its generator is fed (along with a random noise vector in its vanilla version) a conditional vector. This conditional vector describes a class that the generator is supposed to generate (e.g. a male who is married in the Adult dataset). In effect, this is the cross-entropy loss of the given condition and the generated result. It makes the neural network ”listen” to the condition it is given.

4. Classification Loss. The classification loss comes from the auxilliary classifier that is trained with the discriminator and generator, as is described in Park et al. in their TableGAN paper. The classificatier is able to able to tell the differences between the generated and real classes. As Zhao et al. state: ”This helps 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.". In this sense, it helps the neural network generate data that is closer to what is seen in reality.

5.4 Training procedure

To implement Stacked CTAB-GAN, we can take the original CTAB-GAN and modify it in a way that it simultaneously trains the first GAN and second GAN in the CTAB-GAN, and when it is done, it runs one sampling operation on the first generator and saves with it all the generated conditional vectors that were used. The loss for the second GAN is not backpropagated to the first GAN. Thus, the training of these GANs happen at the same time but are completely separated from a weight-tuning perspective. The difficulty lies in creating the condition in which the second GAN can actually be trained to correct the mistakes of the first GAN.

In case of training them separately, the first GAN is first fully trained. Then the first GAN outputs a dataset along with the conditional vectors that were used with it. This dataset as well as the coniditional vectors are then used as input for the second GAN, which trains on it as if it was fed the a random noise vector. As can also be seen in the Empirical section, this method shows sub-optimal results as compared to training them together.

The sizes of the convolutional GANs, which is the case for CTAB-GAN, are dynamically determined by the sizes of the datasets, as is described by Radford et al. in their DCGAN paper. This is the same as the algorithm that was proposed by Zhao et al.

Figure 2 and 3 describe the differences in implementation of the stacked architecture.

It can be seen from figure 2 that the first GAN (orange) gets fully trained, then outputs a single dataset (orange), along with its conditional vectors (blue). This then used by the second GAN (green) and used to produce a second synthesized dataset (green).

Similarly, It can be seen from figure 3 that the first GAN (orange) gets trained together with the second GAN (green). During training, the sampling that is done for feeding the discriminator(orange), is also used for feeding the second GAN (green) along with the conditional vector (blue).

When both GANs are fully trained, the discriminators are discarded and the generators are chained together, practically forming a single generation unit, that can take in a conditional vector and produce a row that resembles the class that the conditional vector describes.

6 Empirical analysis

The empirical analysis part of this research is analogous to part of the method in the CTAB-GAN paper. As cited from Zhao et al.: "To show the efficacy... we select four commonly used machine learning datasets, and compare with... state-of-the-art GAN based tabular data generators. We evaluate the effectiveness... in terms of the resulting ML utility, statistical similarity to the real data, and privacy distance." [1]

6.1 Evaluation Metrics

The metrics are also taken from the ones used by Zhao et al. as this paper uses Zhao et al. as its baseline. The three metrics described below rate the data synthesis quality based on accuracy, as measured by ML Utility and Statistical Similarity and the privacy preservability to measure the extent to which the synthesized data can potentially leak information about an individual. These two types of measures are important as they are conflicting by nature. An ideal synthesizer maximizes accuracy while preserving privacy to the point that no single person's information is traceable.

Machine learning utility
ML utility is quantified by the same five models that are used by Zhao et al., namely: decision tree classifier, linear support-vector-machine, random forest classifier, multionomial logistic regresion and a multi-layer perceptron (MLP). The max-depth of the decision tree and random forest is 28, and 128 nodes for the MLP. Similar to the CTAB-GAN paper, the hyperparameters and models are fixed across the different datasets. The goal of the design is to minimize the the ML utility, as this means the model distribution is well-aligned with the true distribution. Figure 3 and 4 describe how the data flows into the ML utility evaluation. In figure the ML Utility is done "as usual" with a train-test split.

Statistical similarity
The evaluation setup for the statistical similarity is also analogous to what is done by Zhao et al., as it consists of the same three metrics that is used in their paper: Jensen-Shannon divergence (JSD) to calculate the difference between distributions of categorical variables, Wasserstein Distance (WD) for measuring the difference between distributions of continuous variables in a stable manner[1] and Difference in pair-wise correlation (Diff. Corr.), where Pearson correlations are calculated between any two continuous columns and Theil uncertainty is used to the correlation between categorical columns.

Privacy preservability
For the privacy preservability (i.e., the extent to which the individuals' identity is protected), we utilize two distance metrics. The first one is Distance to Closest Record (DCR), which is the smallest distance between any synthetic data point to any real data point and Neighbour Distance Ratio (NNDR), which measures the ratio between the nearest neighbour and second neighbour. If this is too low, this may cause exposure of an individual's personal information to the eventual data consumer, which is generally an unwanted results in terms of privacy regulation.

6.2 Experimental set up

Datasets. The same datasets that were used in the CTAB-GAN are also used in this experiment, namely the **Adult and Covertype** from the UCI machine learning repository[1] and the **Credit** and **Loan** from Kaggle[2]. Similar to the

[1] http://archive.ics.uci.edu/ml/datasets.php

[2] https://www.kaggle.com/Bmlg-ulb/creditcardfraud,itsmesunil/bank-loan-modelling

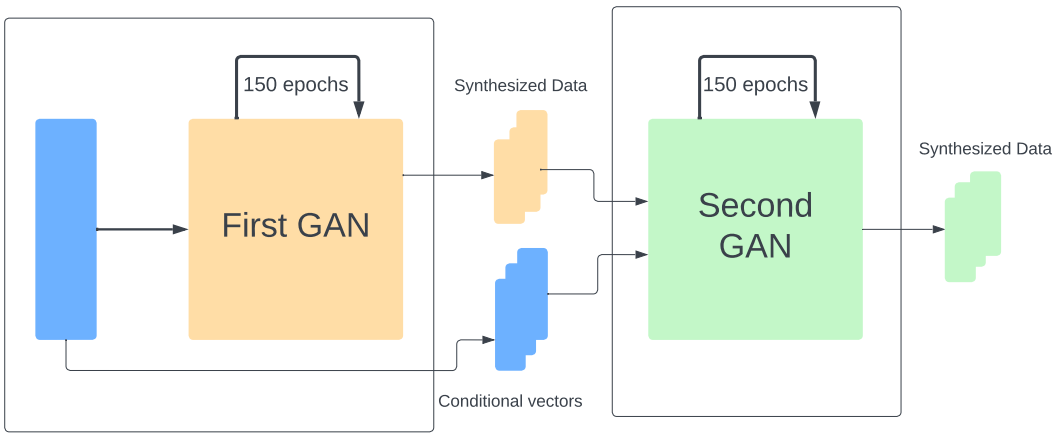

Figure 2: Stacked CTAB-GAN trained separately

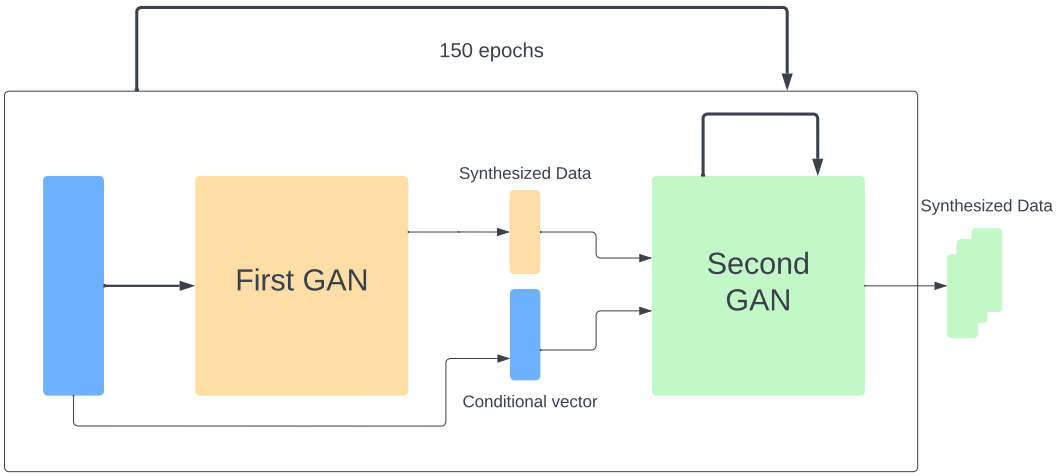

Figure 3: Stacked CTAB-GAN trained together

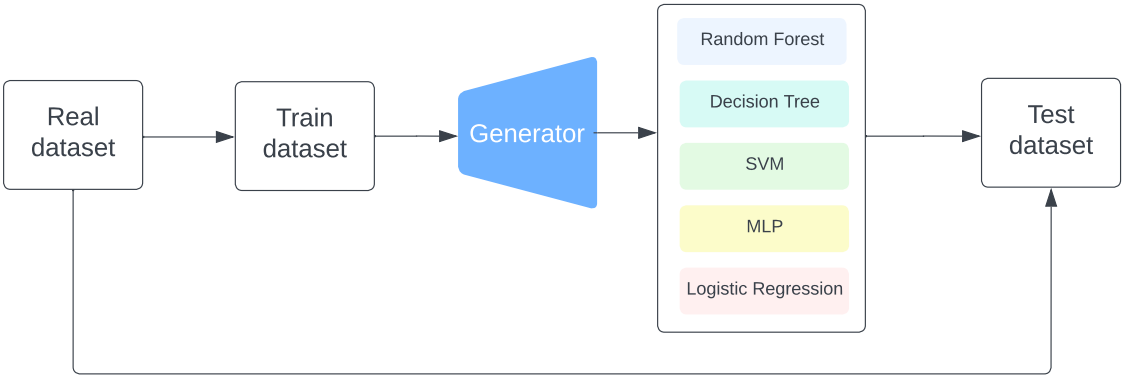

Figure 4: ML Utility for real data

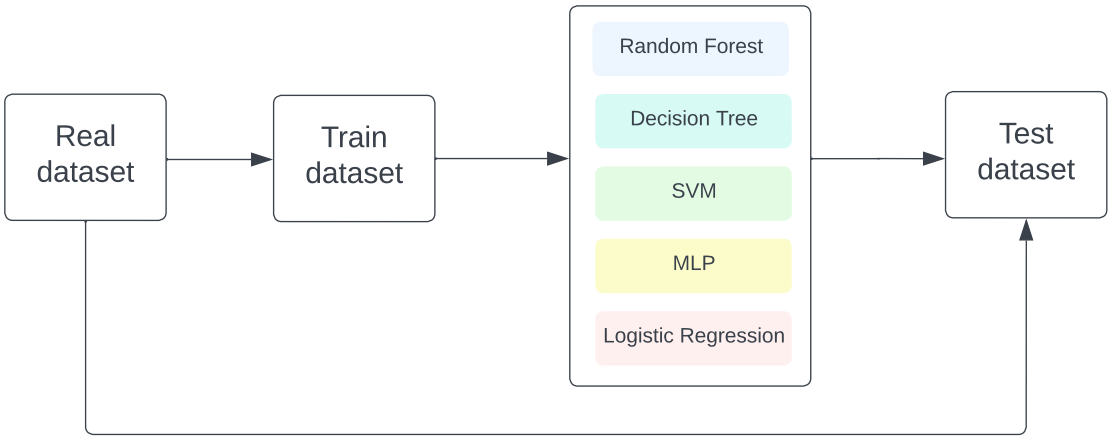

Figure 5: ML Utility for synthesized data

CTAB-GAN and CT-GAN we limit the samples taken from the datasets at 50K and also further assume the the user knows the data type of each variable. [6] The single baseline used for measuring the performance of Stacked CTAB-GAN is CTAB-GAN, as this the objective of this research was to make an improvement on the ”vanilla” CTAB-GAN.

Baselines. This paper only compares the results with CTAB-GAN, as the intent of the research is to improve on this architecture. The other papers that also propose solutions for synthesizing tabular data, are left out.

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.

6.3 Results analysis

The results analysis is done by looking at the metrics that were defined, and comparing them with the results gained from CTAB-GAN. These results show how the new architecture performs when compared to CTAB-GAN, on the three different axis that were defined. As will become clear, there are points that gain from stacking CTAB-GAN, and points that lose out.

6.4 Difference between training together and separate training

Apart from training the two layers of the Stacked CTAB-GAN together, an attempt is also made at training the the GANs separately. In short, the separate training resulted in significantly lower performance, it is also left out for that reason. It does not show any good results that can be used for improving CTAB-GAN.

6.5 Using a full CTAB-GAN for the second layer

The first attempt at improving the original CTAB-GAN was by using a full CTAB-GAN to be chained in the second layer. The idea was that it would be able to use vector of substance to generate from - namely the image that was generated in the first layer as opposed to a random noise vector that is fed to the CTBA-GAN. This had a sub-optimal effect, as can be seen in Table 1 and 2. This result let us re-iterate over the architecture and led us to using a fully connected generator (FCG) for the second GAN.

6.6 Using a fully connected generator in the second stage GAN

The idea behind using a fully connected generator as opposed to a second CTAB-GAN generator, which is CNN-based, is gaining another ”angle of correction”, so that it can detect defects in the generated record that the first layer was not able to handle correctly. As can be seen from Table 1 and 2, the results are better for the fully connected layer as opposed to the double CTAB-GAN. This may have different reasons, of which the following could be one: the architectural capacity of the original CTAB-GAN may be limited to certain inaccuracies - using a second CTAB-GAN for this does not help eliminate the ”overseen” parts of the data synthesis. Using a fully connected generator can perhaps circumvent this and ”fill in the gaps” that the original CTAB-GAN left out. The discriminator and classifier are exactly the same as the original CTAB-GAN. The DCR and NNDR scores are also better for the FCG, which means that the better results do not come at the cost of privacy.

7 Conclusion

There is a multitude of methods in which CTAB-GAN can extended to with progressive learning. Training the first and second layer separately does not yield the result of better synthesis, and in contrast, to achieve optimal results the multiple layers must be trained together. There are strong indications that stacking multiple original CTAB-GAN does not yield better results either. However, by applying changes to the generator for the second layer, there can be a relative gain than to a single CTAB-GAN. The likely reason for this is that the second generator can have a different ”angle of

ML Utility Results			
Metrics	CTAB-GAN	Stacked CTAB-GAN (FCG)	Stacked CTAB-GAN
Accuracy	**6.89%**	8.14%	11.28%
F1-score	0.124	**0.116**	0.197
AUC	0.165	**0.147**	0.231

Table 1: Average Ml Utility results for Adult, Credit, Covtype and Loan datasets.

Statistical Similarity Results			
Metrics	CTAB-GAN	Stacked CTAB-GAN (FCG)	Stacked CTAB-GAN
Average WD	0.031	**0.028**	0.033
Average JSD	0.068	**0.051**	0.140
Correlation Distance	**1.911**	5.655	3.032

Table 2: Average Statistical Similarity Results for Adult, Credit, Covtype and Loan datasets.

Privacy Preservability Results			
Metrics	CTAB-GAN	Stacked CTAB-GAN (FCG)	Stacked CTAB-GAN
DCR between R&S	1.126	1.400	1.409
DCR of R	0.383	0.382	0.383
DCR of S	0.982	1.266	0.777
NNDR between R&S	0.710	0.757	0.800
NNDR of R	0.439	0.451	0.439
NNDR of S	0.672	0.608	0.482

Table 3: Average Statistical Similarity Results for Adult, Credit, Covtype and Loan datasets.

attack", which allows it to rectify defects of the first CTAB-GAN. In short, it is possible to improve synthesis quality, as measured by ML Utility and SSD, by extending CTAB-GAN and it can do so under the same privacy budgets . However, the constraint is that the generator and/or discriminator is constructed in a way that it discerns itself from the original CTAB-GAN generator.

8 Responsible research

The practice and sharing of synthesized data brings responsibilities that are not inherent to sharing real data. With real data there is little ambiguity whether data is personally identifiable and capture reality - its potential bad practices can be clearly warded off. With synthesized data this is different.

8.1 Privacy first

Privacy should be central to all data synthesis that is works on data generated by humans. The privacy metrics shown in this research presents only a fraction of what should be possible for quantification of privacy. The nature of privacy preservability is that it is conflicting with data synthesis accuracy, which indicates that there may be a conflict of interests too in human organizations, e.g. managers pushing for higher accuracy regardless of privacy. Researchers of these techniques should be wary of these potential conflicts and design architectures that can prevent breaches of privacy either through measurability or restrictions.

8.2 Data quality assurance

Another problem inherent to data synthesis is data synthesis quality. In the case of synthesized data exchange between parties, there is no way for the receiver to validate the quality of synthesized data, without (partially) seeing the original dataset. This causes a problem of trust and potentially another conflict of interest in and in between human organization. It is the responsibility of the researcher to find a method in which the quality of data can be assured and is not damaged by, for example, for-profit incentives. This method could be a technical way to verify that the original dataset is actually represented, or another social mechanism must be invented in order to ensure adequate data quality.

References

[1] Z. Zhao, A. Kunar, H. V. der Scheer, R. Birke, and L. Y. Chen, "CTAB-GAN: Effective Table Data Synthesizing," 2021.

[2] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," 2019.

[3] W. R. Tan, C. S. Chan, H. E. Aguirre, and K. Tanaka, "ARTGAN: ARTWORK SYNTHESIS WITH CONDITIONAL CATEGORICAL GANS," 2017.

[4] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling," 2017.

[5] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, "Data Synthesis based on Generative Adversarial Networks," 2018.

[6] L. Xu, M. Skoularidou, A. Cuesta-Infant, and K. Veeramachaneni, "Modeling Tabular Data using Conditional GAN," 2019.

[7] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, and X. H. andDimitris Metaxas, "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks," 2017.

[8] O. Mendez-Lucio, B. Baillif, D.-A. Clevert, and D. R. . J. Wichard, "De novo generation of hit-like molecules from gene expression signatures using artificial intelligence," 2020.

[9] J. Engelmann and S. Lessman, "Conditional Wasserstein GAN-based Oversampling of Tabular Data for Imbalanced Learning," 2020.