MEG: Masked Ensemble Tabular Data Generator

Yishuo Zhang[+]
School of I.T.
Deakin University
zhangyis@deakin.edu.au

Nayyar A. Zaidi[+*]
School of I.T.
Deakin University
nayyar.zaidi@deakin.edu.au

Gang Li
School of I.T.
Deakin University
gang.li@deakin.edu.au

Wray Buntine
College of Engineering & CS
VinUniversity
wray@buntine.org

***Abstract*—Tabular data generation has seen renewed interest with the advent of Generative Adversarial Networks (GAN). Recently, it has been shown that one can use a Bayesian network as either a generator or a discriminator in the GAN framework, resulting in an algorithm known as GANBLR. It has been shown that GANBLR gives state of the art results for tabular data generation. However, the model has one limitation. It uses class attributes during model training. For example, a supervised Bayesian network is needed as a generator at training time. This makes GANBLR inapplicable for cases where we do not have access to class information. Addressing this shortcoming of GANBLR has been the main motivation of this work. In this work, we have proposed a new model of tabular data generation – Masked Ensemble Tabular Generator (MEG), which does not require class labels to generate tabular data. The proposed models rely on a novel strategy of using a collection of Bayesian networks as part of the generator, and relies on masking operations to train the generator efficiently. It also uses a group-based similarity measure to adjust the number of samples generated from each Bayesian network in the collection. We perform extensive experiments on a variety of datasets and demonstrate that MEG not only outperforms baselines that do not have class information during training, such as CTGAN and TVAE, but also outperforms baselines that provide access to class information during training, such as TableGAN and CtabGAN methods. It has almost similar performance in terms of machine learning utility to GANBLR, and of course is greatly advantaged by being truly unsupervised in nature. We highlight this by demonstrating its applicability to a clustering task. We also investigate the privacy preserving capabilities of MEG and demonstrate its superior performance compared to other baselines.**

***Index Terms*—Generative Adversarial Network, Tabular Data, Naive Bayes, Logistic Regression, Bayesian Networks**

I. Introduction

The last few years have seen significant breakthroughs in the world of artificial data generation. Many of these breakthroughs have been due to advances in the development of Generative Adversarial Networks (GAN), a two-part model consisting of a generator and a discriminator that learns to optimise its parameters by optimising an adversarial loss function. GAN-based models have shown remarkable success in generating structured data such as images and text [1], [2], [3]. Their application to tabular data generation has been fraught with challenges. Of course, the main challenge stems from the fact that there is no explicit structure among the input data features to be exploited by the convolutional layers [4], [5], leaving it to the dense layers to develop features to capture the correlation between features. Note that tabular datasets consist mainly of categorical and numerical features. How to seamlessly handle these different feature types in GAN model is not trivial. Recently, it has been shown that for tabular datasets, one can rely on Bayesian networks as generators or discriminators. For example, naive Bayes can be used as the generator and its discriminative equivalent – logistic regression – as the discriminator. The resulting model – GANBLR, has been shown to achieve state-of-the-art results for data generation on a variety of tabular datasets [6]. However, the model has a limitation. It uses the class label during the generation process. Note that the use of the class label in data generation is common in various tabular data generation models, such as TableGAN, CtabGAN, etc [4], [7]. This is because most of the artificially generated data will be used in supervised classification settings and will be expected to produce a model with similar performance (accuracy) to the model trained on the original data [8]. **What if the class information is not available at the data generation stage? Can we still train a data generation model that can generate high quality data without knowing the class labels?**[1] **Investigating this question has been the main motivation for the research conducted in this paper.**

At the heart of GANBLR are Bayesian Networks (e.g., k-dependency Bayesian Networks in [6], and standard Bayesian Networks in [9]). Bayesian networks involve structure and parameter learning. Structure learning determines which attribute or attributes make up the parent set of an attribute. By default, the class attribute is considered to be the parent of each attribute and is therefore excluded from the structure learning stages. For example, a TAN-based Bayesian network will learn the structure (parent-child relationships) based on pair-wise mutual information and minimum spanning tree connecting the attributes – and later add class to be an additional parent of each attribute [10]. Alternatively, we can learn the structure based on a heuristic search technique (e.g. hill climbing) that optimises objectives such as log-likelihood. In this case we have two options. First, one can exclude the class during structure learning, and once the structure is learned, add it as a parent of each attribute. Secondly, the attribute class can be considered as any other attribute and a structure is learned. Standard classification tasks use the first strategy – note that the

* Corresponding author: Nayyar A. Zaidi
+ Equal Contribution

[1]Note that the class labels will be present at evaluation time, i.e. the data will still be used for the supervised learning task. We consider a scenario where the class information is missing during the data generation phase.

second is the unsupervised strategy, as it does not distinguish the class label during structure learning. One can use this version of structure learning of Bayesian networks in GANBLR model to address the issue of missing classes. However, this strategy may be flawed. Note that the identification of the class variable provides implicit information that the class variable is correlated (or connected) with all other features. With unsupervised structure learning in Bayesian networks, there is no guarantee that all features will be selected to be in the Markov blanket of the class. This can lead to the elimination of some important features that would otherwise have been useful indicators for class prediction.

To incorporate Bayesian networks without class information into a GANBLR framework, we have introduced a novel formulation called **M**asked **E**nsemble tabular data **G**enerator – **MEG**, which is based on a collection (or ensemble) of Bayesian networks. As we discussed earlier, a Bayesian network such as TAN or NB could learn the structure based on mutual information (or assuming independence), and later add the class as the parent of each attribute. In MEG, each attribute (including class) takes turns to be the parent, and a separate structure is learned. This means that if we have n variables (including the class attribute), we will learn n structures. MEG uses an effective masking technique to implement only one structure to enable the functionality of multiple structures - and is therefore called *masked* generator. It can be seen that since no prior class information is required, and multiple structures are learned – it is *ensemble* in practice. Finally, it is based on a collection of Bayesian networks used in an adversarial network, so we characterise it as a variant of GANBLR.

In MEG, during adversarial learning, each component of the generator takes turns to generate the samples – and a discriminator is trained to determine whether the generated sample is synthetic (fake) or original. We will discuss how MEG differs from typical GAN-based models in that it relies on group-wise similarity between generated and original samples, in addition to learning to classify fake versus original. This is because, since we have multiple generators producing different samples, – MEG uses the similarity between generated and original data to weight the contribution of each structure in producing the final output. The group similarity loss is used as a weight to control the size of the synthetic sample from the generator. In other words, the better the quality of the synthetic sample generated by a structure – where quality is measured by some metric loss – the lower the loss, the higher the weight learned for that structure, so that more samples are generated from that structure.

Let us summarize the contributions of this work:

- We propose a novel model of tabular data generation that does not require class labels, namely MEG – which uses a collection of Bayesian networks as generators, incorporating a group-based metric loss in the discriminator in addition to the typical fake vs. original discrimination loss. The generators in MEG work with the the discriminator to form an adversarial framework to improve the quality of to improve the quality of synthetic tabular data.
- We compare MEG with existing SOTA GAN models on 10 public tabular datasets. The results show that MEG outperforms in terms of machine learning utility, and comparable results in terms of statistical similarity and related measures.
- We demonstrate the applicability of MEG for unsupervised tasks such as clustering, and discuss its application to privacy preserving scenario.

The rest of the paper is organized as follows. We will discuss related work in Section II. The details of MEG are provided in Section III. We provide an extensive experimental analysis in Section IV. We conclude in Section V with pointers to future works.

II. Related Work

In this section, we will discuss existing GAN-based models for tabular synthetic data generation. Current research into the GAN models for tabular synthetic data generation has taken three directions. The first direction uses the vanilla GAN structure, while the second direction uses the conditional models based on the conditional GAN structure. The last direction is unique and uses the Bayesian model with GAN structure, which can provide excellent interpretation as well as SOTA performance. In the following, we will discuss these three directions in detail.

A. Vanilla GAN-based Tabular Generation

This stream of research is based on the foundational work of [11], in which a random noise (generated from a predetermined distribution) is used as an input to the generator. The generator uses the input to approximate a real data distribution with the encoder and decoder model. The output of the generator is the synthetic data generated. The discriminator uses this synthetic data and the real data to train a classifier to distinguish the synthetic data from the real data. There are four notable works that use this vanilla GAN strategy, and we will discuss them below. Note that we will use the term *vanilla* GAN to refer to the model of [11]. Of course, almost all works using GAN are variants of [11].

The MedGAN [7] model is one of the earliest works to use the auto-encoder architecture as the structure of the generator. The model is capable of generating both categorical and numerical features that are needed to generate authentic electronic medical records. The training in the MedGAN model uses mini-batch averaging to address the mode collapse problem. In addition, batch normalisation with short-cut links is used.

The CrGAN model [12] is designed to generate *Passenger Name Records* (PNR) data for the airline industry. The PNR data includes details of the passenger's personal information such as name, date of birth, booked travel information, flight information, payment details, etc, flight information, payment details, etc. Note that PNR data can consist of both categorical and numerical features that may have missing values. To generate features with missing values using the normal vanilla GAN structure can be challenging. To address this issue, the CrGAN model proposes both categorical feature embedding

and a cross-network architecture. It is shown that the CrGAN model can generate high quality PNR data.

A convolutional neural network is used as a generator of TableGAN [4] and an information loss is used instead. It is shown that TableGAN not only ensures a high machine learning utility, but also claims to preserve the privacy of the data. Note PATEGAN [13] is another notable work, similar to TableGAN, that is designed to prevent privacy attacks. [2].

The vanilla GAN-based tabular generation models mentioned above are effective in different domains. Most of the vanilla GAN-based tabular generation models are unsupervised, which means that they do not need the class to train the generator and discriminator. However the TableGAN which is one of the best performing vanilla GAN-based tabular generation models, is supervised. The TableGAN has the classifier in its framework and requires ground truth (label feature) from the original data.

During the empirical evaluation in section IV, we will use TableGAN as a representative of all vanilla GAN-based methods due to its superior performance.

B. Conditional GAN based Tabular Generation

The conditional GAN-based tabular data generation models use a conditional vector to specify the particular feature value or class label to be generated. Notable work in this stream of research is that of CW-GAN [14], which has been shown to outperform competing methods on various data generation tasks. There are three different loss functions used in the CW-GAN model. The first loss is the *Wasserstein Distance Loss*, which is calculated between the synthetic and real data. The second loss is the gradient penalty, which can be used to regularise the model complexity of the discriminator model. The last loss is the Auxiliary Classifier Loss, which can be used to assist the generator in generating synthetic data belonging to the specified class.

Three well-known models for tabular data generation, such as CTGAN [15], TVAE [15] and CtabGAN [7] also fall into this research stream. CTGAN and TVAE use mode-specific normalisation and training-by-sampling to produce higher quality synthetic datasets with both categorical and numerical features. Inspired by *Gaussian Mixture Model*, mode-specific normalisation first computes the modes of a numerical feature. The mean and standard deviation of each of the modes are then taken. The numerical feature values are then normalised by the associated mean and standard deviation. The newly obtained normalised value is concatenated with the categorical features together to represent the input for CTGAN and TVAE models. The training-by-sampling strategy is the key component of the CTGAN and TVAE generator. It ensures that the instances of the minor class have a similar chance of being sampled as the instances of the major class. The TVAE differs from CTGAN only in its network structure, which uses VAE as the structure and CTGAN uses the fully connected neural networks. CtabGAN has a very similar mechanism for the process of generating synthetic data sets. However, in the CtabGAN framework there is an auxiliary classifier that uses the training data including the label column to improve the quality of the generation. The CtabGAN framework has better performance than CTGAN in terms of machine learning utility. The conditional GAN based generator such as CtabGAN can generate synthetic data with certain conditions such as feature value, however, it requires the specification of a class label and is therefore supervised in nature.

During empirical evaluation in Section IV, we will use CTGAN, TVAE and CtabGAN as a representative of all conditional GAN-based methods.

C. Generative models inspired by Bayesian Network

The current SOTA model for tabular data generation is GANBLR, which trains a Bayesian network (K-dependent Bayesian Network (KDB)) as the generator and a logistic regression (or discriminative KDB) as the discriminator. The generator learns to optimize the conditional log-likelihood of the feature towards the label. Since the generator from GANBLR is a Bayesian network whose parameters constitutes of probabilities, therefore, one can interpret the tabular data generation at both local and global levels. It has been shown that GANBLR provides excellent machine learning performance, however, it has the limitation of generating categorical data only. Another framework under this direction is GANBLR++, which is used to generate both numerical and categorical tabular data [9]. As we discussed earlier, GANBLR and its variants make use of class information, and can not be used in settings where class information is not available.

During empirical evaluation in Section IV, we will use GANBLR as a representative of generative models inspired from Bayesian networks.

III. Method

In this section, we will take a step back and introduce some preliminaries that will serve as a foundation. Later, we will delve into the details of our proposed method MEG.

A. Preliminary and Notations

We will borrow most of the terminology from [6], [16]. We denote the generative model (generator) as $\mathbf{G}$, and the discriminative model (discriminator) as $\mathbf{D}$. The real (or original) dataset is denoted as $\mathcal{D}_{\text{data}} = [(X_g, Y_g)]$, where $X_g = [\mathbf{x}^1, \mathbf{x}^2.., \mathbf{x}^m]$, where $\mathbf{x}^i \in \mathcal{R}^n$, i.e., data with a total of m instances each having n features. Similarly, $Y_g = [y^1, y^2.., y^m]$, where $y^i \in \mathcal{R}^1$, representing corresponding class labels. The term g in the notation makes it explicit that the dataset is to be processed by the generator model – $\mathbf{G}$. We denote the real data distribution as $\mathrm{P}_{\text{data}}(X_g, Y_g)$ or $\mathrm{P}_{\text{data}}(\cdot)$ from which $\mathcal{D}_{\text{data}}$ is generated.

In the GAN formulation, $\mathbf{G}$ is trained to approximate the real data distribution $\mathrm{P}_{\text{data}}(.)$ with some random (noise) input. We denote the random input data as: $\mathcal{Z} = [\mathbf{z}^1, ..., \mathbf{z}^m]$. And the distribution generating $\mathcal{Z}$ as $\mathrm{P}_{\mathcal{Z}}(\mathcal{Z})$ or $\mathrm{P}_{\mathcal{Z}}(.)$. The synthetic

[2]A privacy attack modifies the private aggregation of teacher ensembles so that the privacy noise added to each individual sample of the tabular data is tightly bounded. On the other hand, the noise is added to the aggregation results of all teachers and is therefore difficult to attack.

dataset is denoted as $\mathcal{S}_{\text{data}} = [\bar{X}_g]$, where $\bar{X}_g = [\bar{\mathbf{x}}^1, ..., \bar{\mathbf{x}}^m]$. Here, $\bar{\mathbf{x}}^i \in \mathcal{R}^n$.

As we know that in GAN formulation the generator produces synthetic data from the noise – in our notation, we express it as: $\mathcal{S}_{\text{data}} \sim \mathbf{G}(\mathcal{Z})$. The discriminator model – $\mathbf{D}$, is trained to discriminate between $\mathcal{D}_{\text{data}}$ and $\mathcal{S}_{\text{data}}$. To do this, an auxiliary label $Y_d = 1$ and $Y_d = 0$ is appended with $\mathcal{D}_{\text{data}}$, and $\mathcal{S}_{\text{data}}$ respectively, indicating whether the sample belongs to original or synthetic data. Formally, the objective function of tabular GAN models leads to solving the min-max adversarial game, which in our notation is expressed as:

$$\max_{\mathbf{D}} \min_{\mathbf{G}} \quad \mathbf{E}_{\mathcal{D}\sim \mathrm{P}_{\text{data}}(.)}[\log \mathbf{D}(\mathcal{D}_{\text{data}})] + \mathbf{E}_{\mathcal{Z}\sim \mathrm{P}_{\mathcal{Z}}(.)}[\log(1 - \mathbf{D}(\underbrace{\mathbf{G}(\mathcal{Z})}_{\mathcal{S}_{\text{data}}}))]. \tag{1}$$

It can be seen that, $\mathbf{G}(\mathcal{Z})$ generates the synthetic dataset samples $\mathcal{S}_{\text{data}}$, and $\mathbf{D}$ attempts to convert the synthetic data to a scalar value representing the probability of it being real or not.

The generator and discriminator in GANBLR are actually Bayesian Networks. Precisely, they are generative-discriminative equivalents of each other – e.g., a naive Bayes as generator and logistic regression as discriminator, or, a KDB for generator and a discriminative KDB as discriminator [17]. The generator in GANBLR has two roles, it not only is used to sample the synthetic data, which is used by the discriminator to decide if it is fake or not, but it also influence the final objective function by doing the classification of the class label. The loss attributed to generator can be defined as:

$$\min_{\boldsymbol{\theta_g}} \quad -\log \overbrace{\mathbf{G}(\mathcal{D}_{\text{data}})}^{\mathrm{P}(y|\mathbf{x})} + \log(1 - \mathbf{D}(\overbrace{\mathcal{S}_{\text{data}}}^{\mathbf{G}(.)})), \tag{2}$$

where dependence on the class attribute y is obvious. For sake of completeness, let us also present the discriminator's objective function, optimized by GANBLR:

$$\max_{\boldsymbol{\theta_d}} \quad \log \overbrace{\mathbf{D}(\mathcal{D}_{\text{data}})}^{\mathrm{P}(\text{fake}|\mathbf{x})} + \log(1 - \mathbf{D}(\overbrace{\mathcal{S}_{\text{data}}}^{\mathbf{G}(.)})). \tag{3}$$

Here, $\mathbf{G}(.)$ denotes the sampling output from the trained generator. It can be seen that class attribute plays the pivotal role in optimizing GANBLR objective function. In the following, we will formulate our proposed MEG algorithm that can generate quality tabular synthetic data even when no class label is present (or specified).

B. Masked Ensemble Generative Tabular Generator – (MEG)

Let us discuss our proposed formulation MEG. We will discuss its salient features by characterizing its components namely:

- Collection of Bayesian networks,
- Masked network,
- Adaptive Weighting with group similarity sampling,

before providing a full algorithm in Section III-C. An architecture of MEG is shown in Figure 1.

1) Collection of Bayesian Networks: It can be seen from Equation 2 that GANBLR relies on a supervised Bayesian network. We discussed the possibility of using an unsupervised Bayesian network in Section I and discussed that it might not be very effective at generating data for supervised downstream tasks such as classification. The problem is that for classification, the class feature should be considered as a parent for all other attributes. One way to deal with this problem in structure learning is to use different structures that enforce that all attributes alternate as the class attribute. MEG uses this approach, i.e. data is generated from different structures, and it is the discriminator's job to choose the structure that generates data that most closely resembles the original data.

Consider an example where we have three independent attributes and one class attribute: $\{X_1, X_2, X_3, Y\}$ – MEG initializes four Bayesian networks, in which each of the four attributes will act as the class (or proxy for the class), as shown in Figure 2. For each structure, MEG relies on the strategy of structure learning such as TAN or KDB. That is, it first learns a structure without the class attribute. Later, the class attribute (proxy class attribute) is associated with each attribute. In this paper, in order to simplify the implementation structure instead, i.e. the attributes are assumed to be independent of the class (or proxy class attribute).

It can be seen that some of the structures might be more useful than the others for a downstream task (e.g. for classification, where the dependent variable is, say, feature X_0, the most suitable structure for producing the data will be the one that includes feature X_0 as a class). We will see that MEG relies on adaptive weighting and eventually learns which is the appropriate structure through an adversarial learning process. In some cases, the structures that have (presumably) incorrect attributes as a class can still generate samples that are closer to samples from the true distribution. MEG will learn to distinguish between different structures and weight them differently, eventually learning to sample from the structure that produces data most similar to the target data. However, the weighting is soft weighting. This means that MEG learns a weight distribution over the structures. For example, a 0.2 weight on a structure suggests that MEG will sample only 20% of the data from that structure – because its synthetic data often does not bear a true resemblance to the original data.

During training (similar to GANBLR), MEG enforces the probability constraints on its parameters, i.e., it is made sure that the parameters learned are actual probabilities. Specifically, for every k'th structure in the generator – following constraint is met: $\sum_{j=1}^{\mathcal{X}_i} \theta_{\mathbf{x}_i^j|x_k,\pi_{\mathbf{x}_i}} = 1$, where $\mathcal{X}_i$ represents the cardinality of i-th feature and $\mathbf{x}_i^j$ denotes the j-th feature value for feature i. And x_k represents the value of class attribute, and $\pi_{\mathbf{x}_i}$ denote the value of the attribute which is the additional parent of attribute i. Additionally, the following constraint is also satisfied:

$$\theta_{\mathbf{x}_i^j|x_k,\pi_{\mathbf{x}_i}} = \frac{\exp\left(\theta_{\mathbf{x}_i^j|x_k,\pi_{\mathbf{x}_i}}\right)}{\sum_{j=1}^{\mathcal{X}_i} \exp\left(\theta_{\mathbf{x}_i^j|x_k,\pi_{\mathbf{x}_i}}\right)}, \tag{4}$$

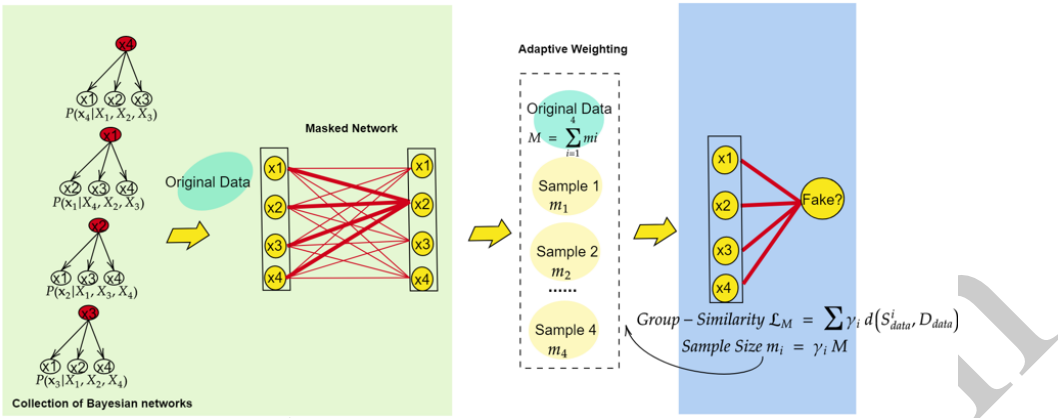

Figure 1: Illustration of MEG architecture.

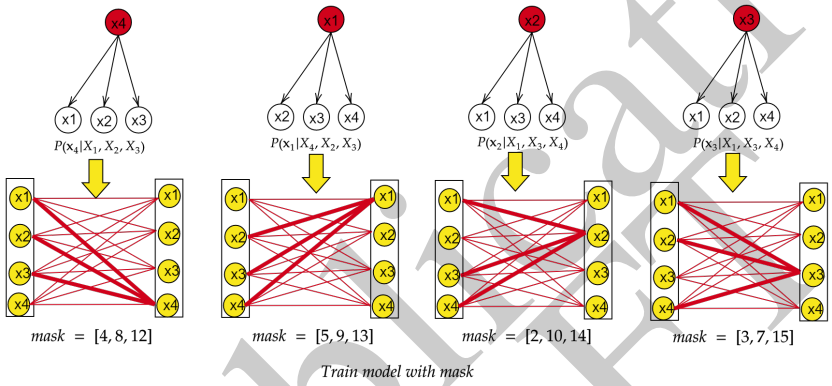

Figure 2: Ensemble Bayesian Network to mask training in Generator. The node of the Bayesian network will be masked by its index.

2) Masked Network: As we discussed earlier, MEG consists of a collection of Bayesian networks, each with its own parameters. In practice, MEG implements a mask-based strategy, i.e. it uses a fully connected dense artificial neural network layer. This is similar to an auto-encoder, where the number of nodes in the first layer is the same as the number of nodes in the last layer, except that there is no intermediate layer. As shown in Figure 2, the masks are specified for each Bayesian network structure. Note that the mask only sets the learnable and fixed parameter settings during learning. Specifically, it specifies which parameters should be learned based on which Bayesian network structure is updated. Let us summarise the generator masking strategy of MEG in the following:

- Giving original data $\mathcal{D}_{\text{data}}$ which has n features – the training of the MEG takes n stages.
- In each training stage n, one of the features $\mathbf{x}_k$ is selected as the label column (we will discuss later in this section, that a loss of form – $\log(\mathrm{P}(\mathbf{x}_k|\mathbf{x}_{\neq k}))$ is optimized by the generator).
- A Bayesian network structure is learned based on strategy such as TAN or NB.
- The parameters which connects the neuron from features $\mathbf{x}_{\neq k}$ to $\mathbf{x}_k$ are set to learnable. All other parameters that are fixed are not learned.
- Once the parameters for k-th structure ($\boldsymbol{\theta}_g^k$) are trained, they are saved and iteration begins for structure belong to the next feature.

At the end of the training, generator network will be parameterized by n set of weights: $[\boldsymbol{\theta}_g^k]_{k=1}^n$ (or denoted as just $\boldsymbol{\theta}_g$).

3) Adaptive Weighting with Group Similarity Sampling: Once the parameters of the generator – $\boldsymbol{\theta_g}$, are trained, it begins its task of generating (sampling) the synthetic data. In this section we will illustrate an adaptive sampling process used by MEG. We will use $\bar{\mathbf{G}}(.)$ to denote the sampling action of the generator. Specifically, we will use the term $\bar{\mathbf{G}}(T)$ for T samples generated by the generator. Similar to the sampling method of GANBLR, forward sampling [16] is used by MEG to generate the synthetic data – $\mathcal{S}_{\text{data}}$. The main difference, of course, is that MEG can sample from n different structures. Not all of these structures are equally important, so the size of the samples generated from each structure must be learned by MEG. Adaptive sampling is done by a specialised component in the architecture of MEG. Unlike the traditional generator and discriminator, MEG has an additional component that is trained to learn the group similarity between the set of samples generated by the generator and the set of samples selected from the original data.

Let us set the size of the total synthetic dataset ($\mathcal{S}_{\text{data}}$)

size over n structures as: $M = \sum_{i=1}^{n} m_i$, where m_i is the sample size for the i-th structure. Note, $\mathcal{S}_{\text{data}}$ is obtained by concatenating sampled data from n structures as:

$$\mathcal{S}_{\text{data}} = [\bar{\mathbf{G}}_1(m_1)\oplus, \bar{\mathbf{G}}_2(m_2)\oplus, \ldots, \oplus\bar{\mathbf{G}}_n(m_n)]. \quad (5)$$

Note, $\bar{\mathbf{G}}_n(m_n)$ denotes data generated by generator utilizing structure n and generating m_n data points.

We calculate the group similarity between synthetic data and the original data as:

$$\mathcal{L}_M = \sum_{i=1}^{n} d(\mathcal{S}^i_{\text{data}}, \mathcal{D}_{\text{data}}). \quad (6)$$

where $\mathcal{S}^i_{\text{data}}$ is the output of $\bar{\mathbf{G}}_i(m_i)$. Before, delving into adaptive weighting, let us discuss our options for computing group-wise similarity. The simplest option is to compute the pair-wise average Euclidean distance or Cosine similarity between $\mathcal{S}^i_{\text{data}}$ and $\mathcal{D}_{\text{data}}$. We, however, have proposed a variant of average similarity. As each sample set $\mathcal{S}^i_{\text{data}}$, comes from structure i, which has a unique class – we compute the pair-wise similarity of samples only belonging to a particular class value. E.g., if attribute i takes $\mathcal{X}_i$ values, and is used as the class in structure generating sample $\mathcal{S}^i_{\text{data}}$, we use the following measure:

$$d(\mathcal{S}^i_{\text{data}}, \mathcal{D}_{\text{data}}) = \frac{1}{\mathcal{X}_i}\sum_{j=1}^{\mathcal{X}_i}\frac{1}{Z}\sum_{q=1}^{m_i}\sum_{r=1}^{|\mathcal{D}_{\text{data}}|}\hat{d}(\mathbf{x}^{(q)}_{\text{S},j}, \mathbf{x}^{(r)}_{\text{O},j}). \quad (7)$$

Here, $\hat{d}(.)$ denotes the Euclidean distance. Z is equal to $m_i \times |\mathcal{D}_{\text{data}}|$. $\mathbf{x}^{(q)}_{\text{S},j}$ denotes q-th data point in synthetic set, where value of feature i equal to j. Similarity, $\mathbf{x}^{(r)}_{\text{O},j}$ denotes denotes r-th data point in original set, where value of feature i equal to j.

In practice, we use a slightly different group-similarity to incorporate weighting. We modify Equation 6 to incorporate weights as:

$$\mathcal{L}_M = \sum_{i=1}^{n}\gamma_i d(\mathcal{S}^i_{\text{data}}, \mathcal{D}_{\text{data}}). \quad (8)$$

It can be seen that we have introduced a term γ_i to weight our group similarity accordingly. Of course γ_i here is $\gamma_i = \frac{\mathcal{L}^i_M}{\mathcal{L}_M}$ and where $\mathcal{L}^i_M = \gamma_i d(\mathcal{S}^i_{\text{data}}, \mathcal{D}_{\text{data}})$. Note, γ_i gives us the recipe to choose the size of sample sets in Equation 5, as we define m_i as $m_i = \gamma_i M$ It can be seen that how MEG automatically adjust the size of sample m_i from each structure, based on the learned parameter γ_i. Note, γ_i is learned as part of generator training, when discriminator is fixed. In the following, we will integrate various components of MEG and formalize its objective function.

4) Typical Discriminator Loss and Proxy Supervised Loss: Let us discuss the objective function of generator in MEG first. As generator constitutes a collection of Bayesian networks in MEG, where each attribute takes turn to be the class – the first term of the objective function of GANBLR in Equation 2 is modified as:

$$-\sum_{i=1}^{n}\log(\mathrm{P}(\mathbf{x}_i|\mathbf{x}_{\neq i})) + \log(1 - \mathbf{D}(\mathcal{S}_{\text{data}})). \quad (9)$$

We call this the proxy supervised loss. As we mentioned earlier, MEG also has an adaptive weighting based on group similarity, where group similarity metric loss is defined Equation 8. We modify generator's objective function to incorporate the group similarity loss as:

$$\min_{\boldsymbol{\theta_g}} \quad -\sum_{i=1}^{n}\log(\mathrm{P}(\mathbf{x}_i|\mathbf{x}_{\neq i})) + \mathcal{L}_M + \log(1 - \mathbf{D}(\mathcal{S}_{\text{data}})). \quad (10)$$

We will be interested to see the relative importance of various component, and therefore, will introduce the two hyper-parameters α and β in generator's objective function as:

$$\min_{\boldsymbol{\theta_g}} \quad -\sum_{i=1}^{n}\log(\mathrm{P}(\mathbf{x}_i|\mathbf{x}_{\neq i})) + \alpha\left[\mathcal{L}_M\right] + \beta\left[\log(1 - \mathbf{D}(\mathcal{S}_{\text{data}}))\right]. \quad (11)$$

Here, hyper-parameter β can be used to control the influence of discriminator, whereas, hyper-parameter α can be used to control the impact of group-similarity loss.

Let us discuss the objective function of the discriminator. MEG has a traditional discriminator component that distinguishes between fake and original datum. Note, the input for discriminator $\mathbf{D}$ is shaped with $[\mathcal{D}_{\text{data}}, Y_d = 1]$ and the synthetic data $[\mathcal{S}_{\text{data}}, Y_d = 0]$. It aims to maximize: $\mathrm{P}(Y_d = 1|\mathcal{D}_{\text{data}}) = \mathbf{D}(\mathcal{D}_{\text{data}})$, and $\mathrm{P}(Y_d = 1|\mathcal{S}_{\text{data}}) = 1 - \mathbf{D}(\mathcal{S}_{\text{data}})$. The discriminator in MEG has similar function as GANBLR but adds the corresponding hyper-parameter β, as it is added in generator's objective function in Equation 11. We modify Equation 3 to write MEG's discriminator's objective function:

$$\max_{\boldsymbol{\theta_d}} \quad \beta\left[\log\mathbf{D}(\mathcal{D}_{\text{data}}) + \log(1 - \mathbf{D}(\mathcal{S}_{\text{data}}))\right], \quad (12)$$

where $\boldsymbol{\theta_d}$ are the parameters of $\mathbf{D}$. It can be seen that one can control the value of β to control the influence of discriminator's network. On one extreme, one can set $\beta = 0$ – this will mean that there is no discriminator, and only sampling from the generator is used to generate samples.

C. MEG – Algorithm

The training of MEG requires to train the generator $\mathbf{G}$'s and the discriminator $\mathbf{D}$'s parameters in turns. We combine Equation 10 and Equation 12 to obtain the complete objective function of MEG as:

$$\min_{\boldsymbol{\theta_g}}\max_{\boldsymbol{\theta_d}} \quad \alpha\left[\mathcal{L}_M\right] + \beta\left[\log\mathbf{D}(\mathcal{D}_{\text{data}}) + \log(1 - \mathbf{D}(\mathcal{S}_{\text{data}}))\right] - \sum_{i=1}^{n}\log(\mathrm{P}(\mathbf{x}_i|\mathbf{x}_{\neq i})) \quad (13)$$

The complete algorithm of MEG is provided in Algorithm 1. In a total of Q iterations (epochs), the input to MEG is used to train $\mathbf{G}$, while fixing the discriminator $\mathbf{D}$. Afterward, the discriminator $\mathbf{D}$ is trained to discriminate between the synthetic and real datasets while fixing Generator's parameters.

Algorithm 1: Algorithm MEG

Input : Original data - $X_g \in \mathcal{R}^n$, β, α
Output : Synthetic data - $\bar{X}_g$

for iteration $q \subset Q$ in training **do**
  Sample m instances $(X_g) \sim \mathrm{P}_{\text{data}}(.)$
  Construct the n mask of MEG
  for version $i \in n$ **do**
    Train the $\boldsymbol{\theta_g}$ for MEG with $\mathbf{x}_i \in (X_g)$
    Obtain $\boldsymbol{\theta_g}$ by optimizing $\mathbf{G}$ via Equation 10 with gradient descent
  Generate $\mathcal{S}_{\text{data}}$ via Equation 5
  for step $t \subset T$ in discriminator $\mathbf{D}$ **do**
    Obtain $\boldsymbol{\theta_d}$ by optimizing $\mathbf{D}$ via Equation 12 with gradient descent
return $\mathcal{S}_{data} \equiv \bar{X}_g$

IV. Experiment and Analysis

A. Experiment setup

1) Datasets: In this work, we use 11 commonly used datasets – 10 from UCI dataset repository and 1 from Kaggle — Loan. All these datasets have a specific dependent variable and a set of independent features. Among them, 3 are large datasets with more than 30K instances (denoted as Large), 4 are medium with less than 30K but greater than 10K instances (denoted as Medium), while the other 4 with less than 10K instances (denoted as Small). Their details are summarized in Table I.

2) Baselines and Evaluation Metric: We compare MEG with GANBLR, CTGAN, TVAE, TableGAN and CtabGAN. Note, among the baselines, only CTGAN and TVAE do not need the specification of the class attribute. While, other baselines that requires the specification of the class attribute during training do not offer a fair comparison, however, the disadvantage lies with our proposed model MEG in this comparison. A performance comparable (or better) to any one these baselines, will demonstrate MEG's efficacy.

All baseline methods are trained with 50 epochs for 3 Large datasets, and 100 epochs for the Medium and Small datasets. Each experiment is repeated 3 times with 2-fold cross validation, and averaged results are reported. It can be seen from Table I that many datasets have number of classes > 2, and hence, we have reported the accuracy (instead of widely used auc measure).

3) Configuration and Running Environment: The Bayesian network in the generator model is constrained to naive Bayes and in the discriminator, it is a simple logistic regression ([17]).

In Section IV-C, we will study the effect of varying the value of β and α. This is to study a) the effect of group similarity loss by varying α and b) to determine the effectiveness of the adversarial (discriminator) component of MEG, by varying β.

MEG is coded in Python 3.7 in Tensorflow 2.5 framework, with 8 core Intel i8 CPU machine with 32 GB RAM memory.

4) Synthetic Data Generation Evaluation: Let us assess the efficiency of MEG in synthetic data generation in this section. To do this, we will evaluate the effectiveness of MEG in terms of:

Machine learning utility (MLU) – which reflects the quality of the synthetic data. Machine learning utility refers to the accuracy obtained from a machine learning model. To fairly evaluate the tabular generators, we use the synthetic data to train several commonly used classifiers such as Random Forest (RM), XGBT, Logistics Regression (LR) and also Mutli-layer Perceptron (MLP), and use the original dataset to test the above-mentioned classifiers (TSTR). For an effective data generation model, a high MLU on TSTR is expected. The TRTR is for training and testing on the original dataset only and used as the indicator comparing with MLU on TSTR.

Statistical similarity – which measures the statistical similarity between the synthetic and the real data. The statistical similarity checks if the distribution of each feature from synthetic data is similar to the original dataset. For an effective data generation model, a high statistical similarity is expected.

Privacy Preserving Capability – which measures the Euclidean distance between any synthetic instance and its closest corresponding neighbor from the original dataset - DCR (Distance to Closest Record). We use DCR to measure whether the synthetic data has too many repeated instances from the original dataset.

Unsupervised Task Evaluation – which reflects the quality of the synthetic data, particularly for unsupervised data, i.e., data without ground truth on the label column. The clustering method such as Kmeans is used to perform the task on both the original data and the synthetic data. The results of the unsupervised task evaluation should have 1) excellent cluster separation and 2) similar cluster centre.

Both, machine learning utility and statistical similarity are standard measures to determine the quality of data generation algorithms [15]. Note, to the best of our knowledge, this is the first work that studies the usefulness of tabular data generation for unsupervised tasks such as clustering. All other evaluation methods, in existing literature, focuses mostly on supervised task such as classification.

B. Results Analysis

1) Results of Machine Learning Utility: The table II shows the averaged accuracy on Large, Medium and Small data sets. It can be seen that MEG outperforms all other basic methods in terms of accuracy based on TSTR except GANBLR. Especially on the small and medium data sets, MEG has a significantly better performance than other baseline methods.

It is interesting to note that MEG's TSTR performance is close to that of GANBLR, while none of the other baseline methods TSTR performance is close to that of GANBLR. It can also be seen that the TSTR performance of MEG is also close to that of TRTR performance, which is the upper limit of accuracy. It is important to note that MEG has almost similar

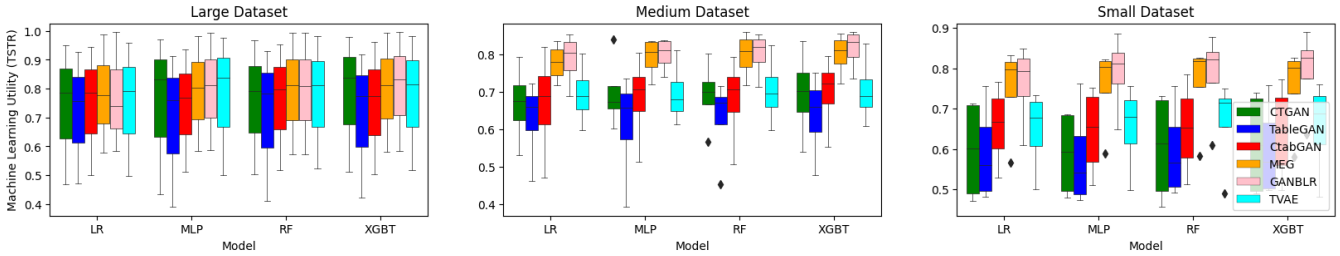

Figure 3: MLU comparison of MEG and competing baseline models on different sized datasets in terms of box-plot (Figure seen best in color).

Table I: Description of datasets.

Dataset	m	n	C	Size
Pokerhand	1M	11	10	Large
Shuttle	58000	9	7	Large
Adult	50000	14	2	Large
Chess	28056	6*	7	Medium
letter_rocog	20000	16	26	Medium
Magic	19020	11	2	Medium
Nursery	12960	8	5	Medium
Sign	6500	8	3	Small
Satellite	6435	12*	7	Small
Loan	5000	13	2	Small
Car	1728	6	2	Small

M denotes million. m, n, C denote the number of instance, features and number of classes, respectively. *Dimension reduction has been applied.

Table II: Average Machine Learning Utility comparison of MEG and competing baseline models on different-sized datasets.

Method	L˜Accuracy%	M˜Accuracy%	S˜Accuracy%
	TRTR		
	80.84%	86.80%	84.62%
	TSTR		
MEG	**78.99%**	**79.19%**	**75.41%**
GANBLR	**79.15%**	**80.10%**	**78.25%**
CTGAN	75.31%	68.77%	60.01%
TVAE	76.61%	69.91%	64.34%
TableGAN	70.48%	62.49%	59.06%
CtabGAN	74.62%	64.05%	67.91%

performance to GANBLR on large and medium size datasets (difference of 0.1% and 0.9% respectively) – it is only on small datasets that MEG has worse performance than GANBLR, where the difference in their performance is 2.84%. We attribute this to the higher number of parameters to learn and train in MEG than GANBLR – of course, more parameters require more data – which is not possible on these small datasets. The same results can be found in table III, which shows detailed TSTR performance for all datasets. These results are very encouraging, as they show that the synthetic data generated by MEG, which does not use class labels, is either of similar or better quality than that generated by any other SOTA data generation algorithm that uses class information.

While Table II provides averaged results, let us look at the distribution of accuracies for different datasets. In Figure 3, we plot the box plots of the (TSTR) accuracy of MEG and its baseline methods on 4 machine learning algorithms. Again, we

Table III: Machine Learning Utility on all datasets in TSTR

Dataset	Model	MEG	GANBLR	CTGAN	TVAE	TableGAN	CtabGAN
Pokerhand	LR	0.579	0.583	0.469	0.498	0.471	0.501
	MLP	0.585	0.588	0.435	0.501	0.39	0.512
	RF	0.572	0.572	0.502	0.522	0.41	0.519
	XGBT	0.580	0.5837	0.511	0.517	0.423	0.502
Shuttle	LR	0.987	0.996	0.951	0.960	0.927	0.945
	MLP	0.982	0.993	0.972	0.976	0.912	0.937
	RF	0.993	0.9931	0.967	0.982	0.929	0.952
	XGBT	0.994	0.996	0.980	0.981	0.92	0.961
Adult	LR	0.778	0.74	0.787	0.791	0.757	0.787
	MLP	0.803	0.812	0.831	0.837	0.761	0.769
	RF	0.812	0.816	0.792	0.812	0.783	0.797
	XGBT	0.813	0.831	0.839	0.816	0.775	0.773
Chess	LR	0.806	0.828	0.693	0.706	0.722	0.717
	MLP	0.835	0.835	0.673	0.699	0.735	0.718
	RF	0.859	0.855	0.701	0.711	0.714	0.722
	XGBT	0.831	0.851	0.723	0.702	0.751	0.737
Letter_recog	LR	0.718	0.689	0.531	0.597	0.462	0.47
	MLP	0.719	0.739	0.601	0.612	0.392	0.512
	RF	0.717	0.713	0.566	0.598	0.452	0.507
	XGBT	0.722	0.736	0.539	0.608	0.478	0.553
Magic	LR	0.752	0.782	0.656	0.671	0.641	0.661
	MLP	0.782	0.789	0.672	0.659	0.632	0.693
	RF	0.783	0.803	0.698	0.681	0.665	0.691
	XGBT	0.792	0.812	0.681	0.675	0.632	0.706
Nursery	LR	0.835	0.852	0.793	0.802	0.678	0.819
	MLP	0.831	0.833	0.839	0.811	0.681	0.805
	RF	0.833	0.835	0.802	0.823	0.676	0.792
	XGBT	0.856	0.859	0.836	0.829	0.688	0.795
Sign	LR	0.567	0.61	0.472	0.501	0.482	0.529
	MLP	0.588	0.649	0.479	0.498	0.473	0.511
	RF	0.583	0.61	0.4563	0.489	0.491	0.512
	XGBT	0.581	0.636	0.489	0.482	0.499	0.498
Satellite	LR	0.832	0.85	0.496	0.642	0.502	0.625
	MLP	0.817	0.887	0.502	0.652	0.492	0.588
	RF	0.826	0.879	0.51	0.717	0.511	0.602
	XGBT	0.815	0.89	0.499	0.656	0.502	0.581
Loan	LR	0.783	0.772	0.706	0.713	0.756	0.766
	MLP	0.789	0.801	0.683	0.708	0.762	0.752
	RF	0.811	0.816	0.731	0.711	0.757	0.786
	XGBT	0.789	0.822	0.739	0.721	0.759	0.773
Car	LR	0.811	0.815	0.713	0.733	0.62	0.711
	MLP	0.823	0.823	0.686	0.757	0.59	0.722
	RF	0.825	0.829	0.719	0.751	0.621	0.705
	XGBT	0.826	0.831	0.721	0.761	0.633	0.711
Average		**0.7776**	0.7915	0.6736	0.7016	0.6343	0.6869

Table IV: Comparison of Statistical Similarity measure of MEG with competing baseline models.

Method	Large		Medium		Small	
	JSD	WD	JSD	WD	JSD	WD
MEG	0.193	0.679	**0.361**	0.682	0.366	1.103
GANBLR	0.161	0.669	0.369	0.690	0.358	0.963
CTGAN	**0.153**	**0.663**	0.367	0.692	0.365	**0.910**
TVAE	0.171	0.681	0.372	0.701	0.363	0.957
TableGAN	0.212	0.731	0.377	**0.663**	**0.351**	1.460
CtabGAN	0.180	0.705	0.372	0.725	0.393	1.025

break the results in Large, Medium and Small datasets. It can be seen that no matter the size of datasets, MEG significantly outperforms all the baseline methods. Especially, for Medium and Large datasets, the performance of MEG is extremely impressive.

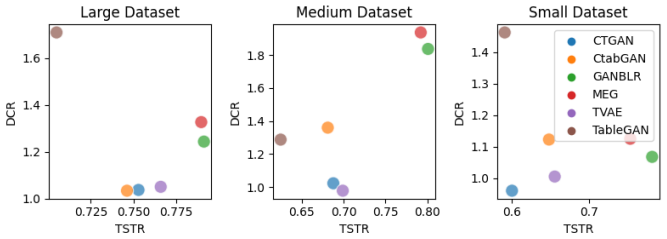

Figure 4: Illustration of the privacy preserving with machine learning utility for different sizes of datasets.

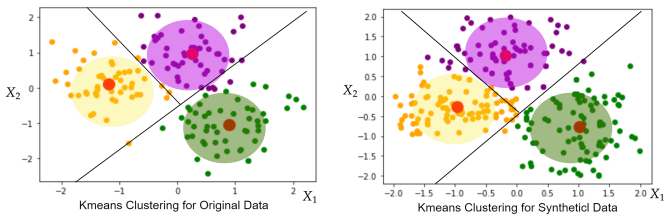

Figure 5: Illustration of the unsupervised evaluation with `Kmeans` for original and synthetic data.

2) Statistical Similarity: To quantitatively measure the statistical similarity between the real and the synthetic datasets, we use **Jensen-Shannon Divergence (JSD)** and **Wasserstein Distance (WD)**. To obtain `JSD` results, for a dataset, each feature from synthetic dataset is measured against the same feature in the real dataset in terms of `JSD`. We repeat the process for all features, and for all datasets, and then, report the averaged result in Table IV – which can be seen as the measure of statistical similarity between synthetic and original dataset.

It can be seen from Table IV, that `MEG` stands out when compared to the other baseline methods. If it is not the best, it is always the second best. Delving into why `MEG` has sub-optimal results in some cases – we believe that this could be due to `MEG`'s generator – collection of Bayesian Network have ability to generate some feature combination that are rarely seen in the real dataset. In the real dataset, the ground truth of the label information is given, however, during the training of `MEG`, each feature will take turns to be a parent. Therefore, statistical similarity perforamance of `MEG` based on `JSD` and `WD` can be impacted.

3) Privacy-preserving Capability Analysis: One of the most important uses of synthetic data is to protect the privacy of training data. data privacy. In many applications, the original training data cannot be directly used for training the model, therefore the synthetic data generated by each of the table generators must have a strong privacy preserving capability. Here, we use a distance-based metric (`DCR`) to quantify the privacy preserving capability of a model. Following the last experiment on statistical similarity, if the mean `DCR` distance between the original and the synthetic data is too large, it simply means that the quality of the generated data is poor. On the other hand, If the distance between the original and synthetic data is too small, it simply means that the quality of the it simply means that there is a risk of sensitive information from the training data. Therefore, the ideal data generation model should make the machine learning utility on `TSTR` as large as possible, and also make the privacy distance `DCR` relatively large.

Figure 4 compares models in terms of `MLU` performance in terms of `TSTR` and privacy preserving performance in terms of `DCR`. It can be seen that both `MEG` and `GANBLR` are better than all other methods, with `MEG` slightly better than `GANBLR` in terms of `DCR`. (red point higher than green point on the y-axis). The slightly better performance of `MEG` compared to `GANBLR` can be attributed to its ensemble collection of Bayesian networks. Of course, it is possible to generate samples that are more diverse and different from the original data.

4) Unsupervised Data Evaluation: Of course, the `MLU` evaluation based on `TSTR` requires the ground truth, e.g., the label information in the data. However, when clustering performance is evaluated, and no ground truth is present, `MLU` evaluation is not applicable. In this experiment, we conduct the unsupervised data evaluation by comparing the clustering performance. There are 150 instances in the training data with 2 features which are artificially generated. Afterward, we use `MEG` to generate the synthetic data, and use the `KMEANS` [18] to cluster the original data and synthetic data with the same amount of clusters ($k = 3$). The results of the clustering are shown in Figure 5. It can be seen from Figure 5, both original data and synthetic data have impressive (yet similar) clustering results. The centre of the `KMEANS` clusters between original data and synthetic data are almost the same. Moreover, the synthetic data generated from `MEG` has been very well separated from the centers. The results from Figure 5 shows that the `MEG` can not only have superior performance on machine learning utility (`MLU`), but also excellent performance on clustering tasks for the unsupervised datasets. Note, `GANBLR` is not applicable in this scenario, as it is supervised in nature.

C. Ablation Study

To illustrate the impact of hyper-parameters α and β on `MEG`, we conducted the ablation studies of `MLU` by changing the configuration of α and β in `MEG` as below:

- $\alpha = 0.1$ **and** $\beta = 0$ As we discussed in Section III, when the $\alpha = 0$ and $\beta = 0$, it means the `MEG` does not include the discriminator part and the generator has a slightly simplified objective function – i.e., it is trained solely by maximizing $\log(\mathrm{P}(\mathbf{x}_i|\mathbf{x}_{\neq i}))$ and group-similarity metric loss ($\mathcal{L}_M$) only, as shown in Algorithm 1.
- $\alpha = 0.001$ **and** $\beta = 1$ In this experiment, `MEG` will have a quite minor impact from the group similarity metric loss ($\mathcal{L}_M$). Therefore, it relies on the discriminator to control the quality of the synthetic data as well as: $\log(\mathrm{P}(\mathbf{x}_i|\mathbf{x}_{\neq i}))$ component of generator.
- $\alpha = 0.01$ **and** $\beta = 1$ In this experiment, `MEG` will have a standard impact from the group similarity metric loss $\mathcal{L}_M$. It is the default setting of the hyper-parameters.

Table V: Ablation study on α and β of MEG.

Hyper-parameters	Impacts	L˜TSTR	M˜TSTR	S˜TSTR
$\alpha = 0.1$ and $\beta = 0$	MEG does not have discriminator	72.15%	73.22%	70.87%
$\alpha = 0.001$ and $\beta = 1$	MEG is barely impacted by metric loss $\mathcal{L}_M$	77.59%	78.31%	73.87%
$\alpha = 0.1$ and $\beta = 1$	Default setting for MEG	**78.99%**	**79.19%**	**75.41%**
$\alpha = 10$ and $\beta = 1$	MEG is largely impacted by metric loss $\mathcal{L}_M$	73.35%	73.12%	70.86%

Table VI: Ablation study on similarity measurement of MEG

Similarity Measurement	L˜TSTR	M˜TSTR	S˜TSTR
Euclidean (default)	78.99%	79.19%	75.41%
Cosine with LSH	78.01%	79.33%	75.43%

$\alpha = 10$ **and** $\beta = 1$ In this experiment, the generator in MEG is largely impacted by the metric loss $\mathcal{L}_M$.

We compare the performance of MEG with each of the above settings using the similar strategy that we used to compare MEG with other competing baselines on the MLU performance. It can be seen from Table V that the default setting ($\beta = 1$ and $\alpha = 0.1$) of MEG has better average performance than all other settings, especially on large datasets. Particularly, Table V also highlights the significance of parameter β in MEG formulation. Without having β, MEG has the worst performance compared to all other settings.

Let us re-visit Table V to see the impact of α. When the value of α is relatively small at 0.001, the results of TSTR have a minor difference with $\alpha = 0.1$ (default setting). Well clearly, it can be seen that higher values of α leads to worse results for small and medium datasets. It is because when $\alpha = 10$, the metric loss $\mathcal{L}_M$ dominates the objective function (Equation 13), which can lead to an inaccurate sampling – as parameter $\boldsymbol{\theta_g}$ may not be learned well.

To find out the performance of MEG on different similarity measures, the ablation study is conducted between the default Euclidean distance and Cosine similarity with local-sensitive-hashing (LSH). It can been seen from Table VI that there is not a huge difference between in the performance of two measures, with approximate measure almost as good as the exact measure. This speaks of the robustness of group-similarity metric used in MEG. Needless to say – Cosine with LSH can convert the quadratic nature of Equation 7 into linear, which can greatly speed-up the data generation process. We have not included the training time results due to space constraints.

In the future work, we are interested to study incorporating higher-order Bayesian networks, i.e., with more than 1 parent (as is the cases in TAN). The applicability of work to fairness enforcement and knowledge-guided and human-in-the-loop learning is to be investigated as well.

V. Conclusion

In this work, we propose a new tabular data generation model MEG, motivated by the need to address the shortcoming of GANBLR in requiring class attributes during training. MEG is based on a collection of Bayesian networks (in which each attribute takes turns to be the class). This is efficiently implemented by a masking operation followed by a group similarity measure between the synthetic and the original data. The group loss is used to derive the number of data points to sample from each network. We demonstrated the effectiveness of MEG by comparing its performance with standard models such as GANBLR, CTGAN, TableGAN and CtabGAN. It was shown that MEG leads to better performance than other baselines (and similar performance to GANBLR). We discussed the suitability of MEG for generating data for unsupervised tasks such as clustering. Given the unsupervised nature of MEG, and given that its performance is almost as good as GANBLR, we characterise MEG as a significant contribution to the field of ever-changing tabular data generation.

References

[1] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, "Cvae-gan: fine-grained image generation through asymmetric training," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2745–2754.

[2] A. Bulat, J. Yang, and G. Tzimiropoulos, "To learn image super-resolution, use a gan to learn how to do image degradation first," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 185–200.

[3] J. Gu, Y. Shen, and B. Zhou, "Image processing using multi-code gan prior," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3012–3021.

[4] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, "Data synthesis based on generative adversarial networks," *arXiv preprint arXiv:1806.03384*, 2018.

[5] A. Kammoun, R. Slama, H. Tabia, T. Ouni, and M. Abid, "Generative adversarial networks for face generation: A survey," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–37, 2022.

[6] Y. Zhang, N. A. Zaidi, J. Zhou, and G. Li, "Ganblr: a tabular data generation model," in *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2021, pp. 181–190.

[7] K. Armanious, C. Jiang, M. Fischer, T. Küstner, T. Hepp, K. Nikolaou, S. Gatidis, and B. Yang, "Medgan: Medical image translation using gans," *Computerized medical imaging and graphics*, vol. 79, p. 101684, 2020.

[8] A. Aggarwal, M. Mittal, and G. Battineni, "Generative adversarial network: An overview of theory and applications," *International Journal of Information Management Data Insights*, vol. 1, no. 1, p. 100004, 2021.

[9] Y. Zhang, N. Zaidi, J. Zhou, and G. Li, "Ganblr++: incorporating capacity to generate numeric attributes and leveraging unrestricted bayesian networks," in *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*. SIAM, 2022, pp. 298–306.

[10] W. Zhang, Y. Liu, C. Dong, and Y. Qiao, "Ranksrgan: Generative adversarial networks with ranker for image super-resolution," in *IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3096–3105.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *arXiv preprint arXiv:1406.2661*, 2014.

[12] A. Mottini, A. Lheritier, and R. Acuna-Agost, "Airline passenger name record generation using generative adversarial networks," *arXiv preprint arXiv:1807.06657*, 2018.

[13] 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.

[14] J. Engelmann and S. Lessmann, "Conditional wasserstein gan-based oversampling of tabular data for imbalanced learning," *Expert Systems with Applications*, vol. 174, p. 114582, 2021.

[15] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," *arXiv preprint arXiv:1907.00503*, 2019.

[16] Y. Zhang, N. Zaidi, J. Zhou, and G. Li, "Interpretable tabular data generation," *Knowledge and Information Systems*, pp. 1–29, 2023.

[17] N. A. Zaidi, G. I. Webb, M. J. Carman, F. Petitjean, W. L. Buntine, M. Hynes, and H. D. Sterck, "Efficient parameter learning of bayesian network classifiers," *Mach. Learn.*, vol. 106, no. 9-10, pp. 1289–1329, 2017.

[18] N. A. Zaidi, D. M. Squire, and D. Suter, "A gradient-based metric learning algorithm for k-nn classifiers," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2010, pp. 194–203.