# GANBLR: A Tabular Data Generation Model

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

Jiahui Zhou
College of Computer Science
Xian Shiyou University
kaezhou@gmail.com

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

***Abstract*—Generative Adversarial Network (GAN) models have shown to be effective in a wide range of machine learning applications, and tabular data generation process has not been an exception. Notably, some state-of-the-art models of tabular data generation, such as CTGAN, TableGan, MedGAN, etc. are based on GAN models. Even though these models have resulted in superiour performance in generating artificial data when trained on a range of datasets, there is a lot of room (and desire) for improvement. Not to mention that existing methods do have some weaknesses other than performance. E.g., the current methods focus only on the performance of the model, and limited emphasis is given to the interpretation of the model. Secondly, the current models operate on raw features only, and hence they fail to exploit any prior knowledge on explicit feature interactions that can be utilized during data generation process. To alleviate the two above-mentioned limitations, in this work, we propose a novel tabular data generation model – *Generative Adversarial Network modelling inspired from Naive Bayes and Logistic Regression's relationship* (GANBLR), which can not only address the interpretation limitation in existing tabular GAN-based models but can provide capability to handle explicit feature interactions. By extensively evaluating on wide range of datasets, we demonstrate GANBLR's superiour performance as well as better interpretable capability (explanation of feature importance in the synthetic generation process) as compared to existing state-of-the-art tabular data generation models.**

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

## I. Introduction

*Generative Adversarial network* (GAN) model and its variants have shown to be effective in a wide range of areas ranging from *Computer vision* [1], *Data Privacy* [2], *Medicine* [3] etc. Typical GAN models [4] constitute of two components: the *generator* learns to produce a synthetic output from the input noise, whereas the *discriminator* learns to distinguish the generator's synthetic data from the real one. Those two components interact based on a game theoretic algorithm such that as the training continues, the generator learns to produce better and better synthetic data, and the discriminator learns to more accurately distinguish between the synthetic and the real data. Typically the generator and the discriminator are modelled as deep *Artificial Neural Networks* (ANN), with the convolution and dense layers. Though, their application to synthetic data generation for computer vision tasks is ground-breaking [5], [6], [7], their application to tabular data generation, has been marred with challenges. Of course, the main challenge stems from the fact that no explicit structure is available among the input data features which can be exploited by the convolutional layers [8], leaving the responsibility to dense layers to engineer features in order to capture the correlation among features. Note, that tabular datasets mainly constitutes of categorical and numeric features. How to seamlessly handle these different feature types is not trivial. Regardless of the challenge, application of GAN to tabular data generation has seen promising results, with models like CTGAN [9], TableGAN [8], MedGAN [3] leading to the state-of-the-art (SOTA) results. E.g, CTGAN generates tabular data via conditional GAN strategy, where the categorical features are regarded as some condition, while using the *Gaussian Mixture Model* estimation for numeric features. It utilizes the *Wasserstein Distance* with gradient penalty to generate the synthetic data. TableGAN, on the other hand, actually uses the convolutional layers in both the generator and the discriminator stages, and introduces an information loss based objective function.

We believe that tabular data generation (especially with GAN models) is still in early days, and there is strong demand for more accurate and far better interpretable data generation models. In this work, we will demonstrate that GAN strategy is indeed effective, but (instead of relying on ANN and its variants) a fundamentally different approach to its data generator and discriminator components can lead to much better results. Before we discuss our formulation, let us discuss some limitations of existing GAN-based tabular data generation models:

- First, effective modelling of feature interactions is critical in many machine learning tasks, and data generation is without exception [10], [11]. Of course, generators and discriminators in vanilla GAN models can capture feature interaction, but this implicit feature interaction will not be interpretable [11]. Also, there is no guarantee that any particular interaction is captured by the model. E.g., we may have some prior knowledge that feature Salary is highly correlated with feature Age — but the generator in vanilla GAN may or may not capture this interaction. Because, it is likely that the model can find some other more useful interactions. This lack of explicit feature interaction modelling is one of the main factors impacting the performance of synthetic data generation. Of course, one solution is to craft the feature interaction manually — but this will be tedious and time consuming.

Corresponding author: Nayyar A. Zaidi
* Equal Contribution

- Secondly, related to the first issue that we discussed, the usage of ANN and variants in existing GAN-based tabular data generation models lead to a generation process that is hard to interpret. For tabular data generation, since the goal is to improve the performance of machine learning tasks, the demand for interpretability is far more crucial.

It can be seen that these two limitations of existing vanilla GANs mainly stem from the fact that their generator component is carved out of a deep ANN (or its variants). Can we utilize a different model as the generator to address these limitations? Answering this question has been the main motivation of this work.

In this work, we propose a radically different formulation, which departs from existing vanilla GAN based data generation – instead of using a deep ANN (or its variants) as the generator (and the discriminator), we utilize the Bayesian Network (BN) model. A BN is a directed acyclic graphical model – in which the training process incorporates learning the structure as well as its associated parameters. By specifying (or learning) the structure, one can explicitly incorporate feature interactions, whereas since its parameters correspond to actual probabilities, the model is interpretable. It can be seen that BN has desirable data generation properties which can address the above-mentioned limitations of existing tabular GAN models. But how can one use BN in the GAN formulation? To answer this question, let us dive deeper in BN.

Typical BN models works with tabular data and maximizes the *log-likelihood* (LL) $\sum_{i=1}^{m} \log \mathrm{P}(y^i, \mathbf{x}^i)$, where $m$ is the number of data points, $\mathbf{x}$ is a vector of independent features with discretized values and $y$ is the dependent target feature. Note, the numeric features of $\mathbf{x}$ are discretized prior to calculating the probabilities in BN models. A BN is an example of a generative model, as one can use it to generate (sample) data once the model is learned. Of course, one can calculate $\mathrm{P}(y^i|\mathbf{x}^i)$ in BN to obtain a classifier. However, the predictive performance of BN classifiers is not generally good, as compared to the models that directly optimizes $\mathrm{P}(y^i|\mathbf{x}^i)$ – also known as the discriminative models. Interestingly, generative models such as BN can be trained by optimizing a discriminative objective function such as the *conditional log-likelihood* (CLL) $\sum_{i=1}^{m} \log \mathrm{P}(y^i|\mathbf{x}^i)$. E.g., a popular example of BN is naive Bayes (NB) classifier, whose discriminative equivalent is the well known *Logistic Regression* (LR) model which of course optimizes the CLL. NB and LR are well-known examples of generative-discriminative equivalent models – in general, one can train any BN by optimizing the CLL – leading to a respective generative-discriminative equivalence. Following the notation in [12], we denote a BN trained by optimizing the CLL as $\mathtt{BN}^d$, and then under this notation, we have $\mathtt{LR} = \mathtt{NB}^d$ [13]. Typically, since structure learning of BN is time consuming – we can resort to simple restricted models such as *Tree-Augmented Naive Bayes* (TAN) or *K-Dependence Bayesian* estimators (KDB), in which the structure learning only takes one or two passes through the data. Now, one can train TAN or KDB by optimizing a discriminative objective function, i.e., CLL – leading to either $\mathtt{TAN}^d$ or $\mathtt{KDB}^d$ formulations.

Although one can sample from a standard BN the samples might not be of good quality – since when optimizing $\mathrm{P}(y, \mathbf{x})$, there is no guarantee that $\mathrm{P}(y|\mathbf{x})$ will be well-calibrated [14]. One solution is to directly sample from $\mathtt{BN}^d$, which optimizes $\mathrm{P}(y|\mathbf{x})$. Well, here the issue is that the parameters are not constrained to be actual probabilities [1]. One solution is to constrain the weights to be the actual probabilities during the training of a discrmiminative objective function. Such weights constraining has been explored for LR and KDB in [12]. Following the naming conventions from existing work in this area, we denote a BN which is learned discriminatively, but constrain weights so that they conform to actual probabilities as $\mathtt{BN}^e$.

We claim that $\mathtt{BN}^e$ is the perfect model to be used as a generator in GAN models. Particularly, it optimizes a discriminative objective function (and hence can be learned end-to-end with back-propagation algorithm), the weights are actual probabilities so you can sample, and importantly you can interpret the model based on the learned weights. One drawback of using BN in GAN models is that, it can only process and hence generate data with categorical attributes only. Note, it has been recently shown that contrary to popular belief, discretization can lead to models that have superiour performance as compared to their numeric counter-parts [15]. Therefore, discretizing numeric attributes, and then sampling using $\mathtt{BN}^e$ can be an effective alternative to directly generating the numeric attributes.

In our proposed formulation of GAN models, we will use typical Bayesian Network models as the generator ($\mathtt{BN}^e$, e.g., $\mathtt{NB}^e$, $\mathtt{TAN}^e$, $\mathtt{KDB}^e$) and their discriminative counter-parts as discriminator ($\mathtt{BN}^d$, e.g., $\mathtt{NB}^d$, $\mathtt{TAN}^d$, $\mathtt{KDB}^d$, etc.). We name this new formulation as: ***G**enerative **A**dversarial Network modelling inspired from **N**aive **B**ayes and **L**ogistic **R**egression's relationship* (GANBLR). However, NB and LR terms in the GANBLR acronym are only figurative to represent broad generative and discriminative learning paradigms. In practice, one can use any generative model as the generator and its corresponding discriminative model as the discriminator.

Let us summarize the contributions of this work:

- We propose a novel model of tabular data generation, namely GANBLR – which uses $\mathtt{BN}^e$ as the generator and $\mathtt{BN}^d$ as the discriminator. Note, the generator in GANBLR cooperates with the discriminator to form an adversarial structure to improve the quality of synthetic tabular data. Note, GANBLR is limited to producing datasets that have categorical attributes only.
- Even though $\mathtt{BN}^e$ has been studied in the context of classification – this is the first work which studies its effectiveness as a data generation technique.
- We compare GANBLR to existing SOTA GAN models on 15 public tabular datasets. The results demonstrate that GANBLR not only outperforms in the **machine learning**

[1] E.g., we can sample a point from naive Bayes, but not from LR.

**utility** [2] and the statistic similarity measured with **Jensen-Shannon Divergence** and **Wasserstein Distance**, but also provides better interpretability.

The rest of the paper is organized as follows. We will discuss related work in Section II. The details of GANBLR is 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

Existing application of GAN models to tabular data generation has followed two directions – the first is based on the vanilla GAN structure, while the second is based on the conditional GAN structure. Let us briefly discuss these two directions.

*1) Vanilla GAN-based Tabular Generation:* This stream of research relies on seminal work of [4], such that the generator takes on some random noise as input and is trained to approximate the real data distribution. The discriminator learns to discriminate the synthetic data from the real data. Several models fall in this category, including MedGAN [3], CrGAN [16], TableGAN [8], PATEGAN [2], etc. Among these, MedGAN utilizes the auto-encoder architecture as the generator to generate both categorical and numerical features. The model has produced some realistic synthetic health record data. On the other hand, CrGAN model was designed to generate the *Passenger Name Records* (PNR), with details of passenger's personal information such as name, date of birth, the reserved trip information, the flight information and the payment details, etc. It uses the Cramer distance in the synthetic data generation for providing the unbiased sample gradients to better handle the gradient descent, and to overcome the local minima during training. TableGAN utilizes the convolutional neural network in the generator part to capture the information of feature at more granular level. The model also adopts the information loss to differentiate the discrepancy between the synthetic and the real data. TableGAN also claimed that the generated synthetic data could have better privacy preserving properties along with excellent machine learning utility. Similar to TableGAN, PATEGAN is designed to prevent the privacy attacks [3]. And the tabular data generated has the differential privacy guarantee to avoid the privacy attack.

Tabular GAN models have shown to be effective in different domains, but still have two critical limitations. First, the models are only proposed for generating binary class datasets, and their suitability to multi-class settings is not clear and well-established. Secondly, these models are not able to generate the synthetic data with a particular (or specified) feature-value. E.g., TableGAN can not generate the credit information for a female card holder who lives in say New York. This is a crucial limitation for imbalanced machine learning datasets such as fraud detection and alleviating it has been the main focus of conditional GAN models, which we discuss next.

*2) Conditional GAN based Tabular Generation:* Conditional GAN-based tabular data generation models relies on a conditional-vector to specify the particular feature value or class label to generate. Notable examples include CW-GAN [17] and the well-known CTGAN [9]. CW-GAN has been shown to get results better than competing methods on credit data generation. The model optimizes three different losses a) *Wasserstein Distance* between the synthetic and the real data; b) the gradient penalty for regularizing the model complexity on the discriminator and c) the auxiliary classifier loss, encouraging the generator to generate samples that recognizably belong to the given class. CTGAN is the current SOTA model for tabular data generation. To better generate both categorical and numeric features it adopts a mode-specific normalization for numeric features and training-by-sampling process in its framework [18], [8], [3]. The mode-specific normalization firstly compute the modes of a numeric feature by *Gaussian Mixture Model*, then calculate the probability of the value coming from each of the mode. Lastly, the value of the numerical feature is represented by the mode category, and also normalized by the mean and standard deviation from the probability density of the mode. Later, the normalized numerical features are concatenated with categorical features to represent as the input for CTGAN. The training-by-sampling strategy in the conditional generator of CTGAN can best utilize the advantage of the condition-vector to balance categorical features via synthetic data generation. For example, the instances of minor class can have less chance to be sampled without the training-by-sampling strategy. However, in training-by-sampling strategy, the conditional vector makes all the categories from discrete attributes to sample evenly, therefore, the synthetic data distribution matches the real data distribution.

## III. Method

Let us start by presenting some preliminary work to be used as a foundation, as well as some notation that is used throughout in this paper. Later, we will delve deep in our proposed algorithm – GANBLR.

### A. Preliminary and Notations

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 $\mathcal{D}_{\text{data}} = [(X_g^k, Y_g)]$, where $X_g^k = [\mathbf{x}^1, \mathbf{x}^2.., \mathbf{x}^m]$, where $\mathbf{x}^i \in \mathcal{R}^n$, i.e., data with a total of $m$ instances each having $n$ independent features, with $k$-order feature interaction present among them. Similarly, $Y_g = [y^1, y^2.., y^m]$, where $y^i \in \mathcal{R}^1$, constituting corresponding class labels. Note, we make the dependence on the order of feature interaction ($k$) explicit in our formulation. Each dataset has a maximum level of interactions present among features, which is denoted by $k$ here. Of course, for a generator to produce samples effectively, it must be able to model these $k$-order interactions present in the data. If we are using a $\text{BN}^e$ model as the generator, we can easily specify $k$, however, if

[2]Machine learning utility means that the synthetic data is used to train the model and then use real data to test the performance.

[3]A privacy attack modifies the private aggregation of teacher Ensembles to allow the tight bound of the privacy noise added on each individual sample of the tabular data.

| Notations | Representation |
|---|---|
| $\mathbf{G}$ | Generator model |
| $\tilde{\mathbf{G}}(\mathcal{D}_{\text{data}})$ | Sampling role of Generator |
| $\bar{\mathbf{G}}(.)$ | Sampling role of Generator |
| $\mathbf{D}$ | Discriminator model |
| n | Number of Features |
| m | Number of Instances |
| k | Order of feature interaction |
| $\mathcal{D}_{\text{data}} = [(X_g^k, Y_g)]$ | Real dataset used in generator |
| $\mathbf{x}^i, y^i$ | Sampled data instance from real dataset |
| $x_i, y, \pi_{x_i}$ | Feature values in the input datum |
| $\mathcal{S}_{\text{data}} = [(\bar{X}_g^k, \bar{Y}_g)]$ | Synthetic dataset |
| $\mathcal{Z} \sim \mathrm{P}_z(.)$ | Random noise input |
| $\mathcal{S}_{\text{data}}$ | Synthetic data generated from generator |
| $[(\mathcal{S}_{\text{data}}, Y_d = 1)]$ | Real data input of discriminator |
| $[(\mathcal{S}_{\text{data}}, Y_d = 0)]$ | Synthetic data input of discriminator |
| BN$^e$ | Bayesian Network as the generator in GANBLR |
| BN$^d$ | Bayesian Network as the discriminator in GANBLR |

Table I: List of symbols used in this work.

a traditional deep ANN is used, modelling of interactions of order-$k$ is more of a trial-and-error practice to determine the right breadth and depth of the generator network. 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 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 dataset is denoted as $\mathcal{S}_{\text{data}} = [(\bar{X}_g^k, \bar{Y}_g)]$, where $\bar{X}_g^k = [\bar{\mathbf{x}}^1, ..., \bar{\mathbf{x}}^m]$, and $\bar{Y}_g = [\bar{y}^1, ..., \bar{y}^m]$. Here, $\bar{\mathbf{x}}^i \in \mathcal{R}^n$ and $\bar{y}^i \in \mathcal{R}^1$. Again, the superscript $k$ denotes that the synthetic data should have the same order of feature interactions as in the original dataset.

As we know that in GAN formulation the generator generates 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, specifying if 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 $\mathbf{D}$ tries to map the synthetic data to a scalar value representing the probability of it being real or not.

In the following, we will discuss how to use BN$^e$ as the generator and BN$^d$ as the discriminator, leading to our GANBLR formulation. List of all the symbols used in this work is given in Table 1.

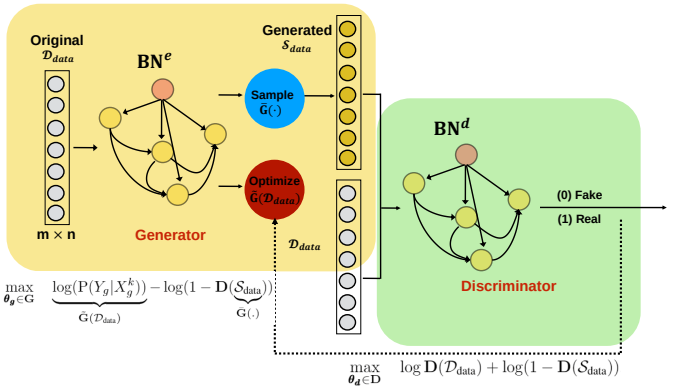



Figure 1: GANBLR Framework.

### B. GANBLR – Components

The generator in our proposed formulation deviates from vanilla GAN as it has two roles to play:

- Its first role is to learn the parameters of BN$^e$. By doing so, it learns the weights by optimizing the discriminative objective function, while fulfilling the probability constraints on the weights. We denote the generator for this training role as $\tilde{\mathbf{G}}$. Note, the input to $\tilde{\mathbf{G}}$ is the original data $\mathcal{D}_{\text{data}}$, i.e., we can write: $\tilde{\mathbf{G}}(\mathcal{D}_{\text{data}})$.
- The second role of the generator is to sample data after the discriminative Bayesian Network BN$^e$ is trained. Since the optimized parameters are conditional probabilities, now we can use BN$^e$ in the generative mode to sample the synthetic data samples $\mathcal{S}_{\text{data}}$. We denote this sampling role as $\bar{\mathbf{G}}$. The input to this role of generator is null – i.e., we can write: $\bar{\mathbf{G}}(.)$. Note, this formulation deviates from existing tabular GAN models, as our generator does not generate from random noise distribution.

The two roles of the generator in GANBLR works seamlessly in an overall adversarial training framework – first, the generator operates in a training role ($\tilde{\mathbf{G}}$) for optimizing its weights discriminatively under some constraints, and then shift its role to sampling ($\bar{\mathbf{G}}$), for synthetic dataset generation. For sake of simplicity, we will use notation $\mathbf{G}$ for generator in cases where the role of generator is cleared from the context.

The discriminator $\mathbf{D}$ in GANBLR is again a Bayesian Network – BN$^d$ (which is trained discriminatively, but no constraints on the weights). It learns to distinguish between $\mathcal{D}_{\text{data}}$ and $\mathcal{S}_{\text{data}}$. The loss from $\mathbf{D}$ is back-propagated to the generator $\mathbf{G}$ for the improvement of the synthetic data generation. Let us delve into the details of each component of GANBLR.

*1) Generator* $\mathbf{G}$*:* Let us establish the form of the generator first. In GANBLR, we recommend to use restricted Bayesian Network model of KDB. Although, any form of Bayesian Network can be used in GANBLR framework, restricted Bayesian Networks have some advantages. First, the structure can be configured easily by specifying the hyper-parameter, i.e., number of parents ($k$). Therefore, GANBLR only focuses on parameter learning given the structure. Secondly, since, a BN

with immoral nodes can lead to non-convex problems according to [19], using a restricted Bayesian Network decreases the chances to obtain immoral nodes. E.g., with $k < 2$, we do not have the problem of immoral nodes. However, with $k \geq 2$, discriminative training of BN can lead to non-convex optimization [4].

The KDB model use the mutual and conditional mutual information to learn the structure. A typical feature interaction in a KDB model includes feature itself, the target feature, and its parent feature(s). As discussed earlier, we wish to train KDB discriminatively with some constraints fulfilled – leading to $\text{KDB}^e$ formulation. However, we will use the term $\text{BN}^e$ instead (for sake of generalization).

The generator in GANBLR optimize following two objective functions:

- Maximizing $\tilde{\mathbf{G}}(\mathcal{D}_{\text{data}})$ which is the conditional log-likelihood of the form: $\log(\text{P}(Y_g|X_g^k))$, and
- Minimizing the loss $\log(1 - \mathbf{D}(\tilde{\mathbf{G}}(.)))$ or $\log(1 - \mathbf{D}(\mathcal{S}_{\text{data}}))$.

Instead of minimizing $\log(1 - \mathbf{D}(\mathcal{S}_{\text{data}}))$ we can maximize: $-\log(1 - \mathbf{D}(\mathcal{S}_{\text{data}}))$ instead, which leads to the objective function for the generator $\mathbf{G}$ as:

$$\max_{\boldsymbol{\theta_g} \in \mathbf{G}} \quad \underbrace{\log(\text{P}(Y_g|X_g^k))}_{\tilde{\mathbf{G}}(\mathcal{D}_{\text{data}})} - \log(1 - \mathbf{D}(\underbrace{\mathcal{S}_{\text{data}}}_{\bar{\mathbf{G}}(.)})). \tag{2}$$

Note, just like vanilla GAN models, $-\log(1 - \mathbf{D}(\mathcal{S}_{\text{data}}))$ part of the objective function is not involved while training the parameters of $\mathbf{G}$, as the discriminator $\mathbf{D}$ is fixed during $\mathbf{G}$'s optimization.

Let us focus on the generator parameters ($\boldsymbol{\theta_g}$) that are to be learned in Equation 2. For this, we write $\log(\text{P}(Y_g|X_g^k))$ as:

$$\begin{aligned} \log \text{P}(Y_g|X_g^k)) = (\log(\theta_y) + \sum_{i=1}^{n} \theta_{x_i|y,\pi_{x_i}} \\ + \log(\sum_{y'}(\theta_{y'} \prod_{i=1}^{n} \theta_{x_i|y',\pi_{x_i}})). \end{aligned} \tag{3}$$

Here $\theta_y$ denotes the weight associated with class prior or can be considered as the intercept. $x_i$ denotes the feature value for $i$-th feature, and $\pi_{x_i}$ denotes the set of feature-values of those features which are feature $i$'s parents. Note, $y$ denotes the class value, also class is the parent of each feature. Since Bayesian Network structure leads to conditional probabilities, our notation is symbolic as we represent a weight in our network as: $\theta_{x_i|y,\pi_{x_i}}$. In practice, GANBLR has a parameter $\theta$ associated with interaction: $x_i, y, \pi_{x_i}$. Note, $\log \sum_{y'}(\theta_{y'} \prod_{i=1}^{n} \theta_{x_i|y',\pi_{x_i}})$ is the normalization term to make sure that $\text{P}(Y_g|X_g^k)$ is between 0 and 1.

Notably, GANBLR enforces the constraints on $\boldsymbol{\theta_g}$, making sure that:

$$\sum_{j=1}^{\mathcal{X}_i} \theta_{x_j|y,\pi_{x_i}} = 1, \tag{4}$$

where $\mathcal{X}_i$ represents the cardinality of feature $i$, and $x_j$ represents $j$-th feature value. Additionally, following constraint is satisfied:

$$\theta_{x_i|y,\pi_{x_i}} = \frac{\exp(\theta_{x_i|y,\pi_{x_i}})}{\sum_{j=1}^{\mathcal{X}_i} \exp(\theta_{x_j|y,\pi_{x_i}})}. \tag{5}$$

Once the $\text{BN}^e$ weights are trained, the second role $\bar{\mathbf{G}}$ of the generator, that is, to generate the data begins. One can generate the synthetic data $\mathcal{S}_{\text{data}}$ by using the forward sampling [20]. One can set the size of synthetic dataset size ($m$), in forward sampling, whereas, the feature interaction order $k$ is expected to be the same as that of generator $\mathbf{G}$'s input, i.e., $\mathcal{D}_{\text{data}}$. The sampling process of generator $\mathbf{G}$ is quite evident, and can be expressed as:

$$\mathcal{S}_{\text{data}} = \bar{\mathbf{G}}(.). \tag{6}$$

*2) Discriminator* $\mathbf{D}$*:* The discriminator $\mathbf{D}$ determines the quality of the synthetic data $\mathcal{S}_{\text{data}}$ during the training, and then of course, back-propagate the error to the generator $\mathbf{G}$. Note, as we discussed, the generator in GANBLR $\mathbf{G}$ gets the loss from the discriminator $\mathbf{D}$ to adjust its weights $\boldsymbol{\theta_g}$. The input for discriminator $\mathbf{D}$ is shaped with $[\mathcal{D}_{\text{data}}, Y_d = 1]$ and the synthetic data $[\mathcal{S}_{\text{data}}, Y_d = 0]$. In GANBLR, the discriminator $\mathbf{D}$ is again a Bayesian Network model ($\text{BN}^d$) trained to optimize the CLL with the hyper-parameter ($k$) same as that of generator's Bayesian Network ($\text{BN}^e$). The training of discriminator $\mathbf{D}$ aims to maximize:

- $\text{P}(Y_d = 1|\mathcal{D}_{\text{data}}) = \mathbf{D}(\mathcal{D}_{\text{data}})$, and
- $\text{P}(Y_d = 1|\mathcal{S}_{\text{data}}) = 1 - \mathbf{D}(\mathcal{S}_{\text{data}})$,

by optimizing the following objective function:

$$\max_{\boldsymbol{\theta_d} \in \mathbf{D}} \quad \log \mathbf{D}(\mathcal{D}_{\text{data}}) + \log(1 - \mathbf{D}(\mathcal{S}_{\text{data}})), \tag{7}$$

where $\boldsymbol{\theta_d}$ are the parameters of $\text{BN}^d$, whereas $\log(1 - \mathbf{D}(\mathcal{S}_{\text{data}}))$ is passed on to the generator $\mathbf{G}$, as in standard vanilla GAN formulation.

### C. GANBLR – Algorithm

The training of GANBLR requires to train the generator $\mathbf{G}$'s and the discriminator $\mathbf{D}$'s parameters in turns. We combine Equation 2 and Equation 7 to obtain the objective function of our GANBLR formulation:

$$\begin{aligned} \max_{\boldsymbol{\theta_d}} \min_{\boldsymbol{\theta_g}} \quad & \log \mathbf{D}(\mathcal{D}_{\text{data}}) + \log(1 - \mathbf{D}(\mathcal{S}_{\text{data}})) \\ & - \log(\text{P}(Y_g|X_g^k)). \end{aligned} \tag{8}$$

The complete algorithm of GANBLR is provided in Algorithm 1, and we provide its architecture in Figure 1. In a total of $Q$ iterations (epochs), the input to GANBLR is used to train $\tilde{\mathbf{G}}$, while fixing the discriminator $\mathbf{D}$. Afterwards, the discriminator $\mathbf{D}$ is trained to discriminate between the synthetic and real datasets.

[4]We show in our experiments that such non-convexity is not a huge issue, as we still get good results with KDB where $k = 2$. Testing with higher values of $k$, as well as with unrestricted BN models has been left as a future work.

**Algorithm 1:** Algorithm GANBLR

**Input** : Original data - $X_g^k, Y_g$
**Output** : Synthetic data - $\bar{X}_g^k, \bar{Y}_g$

**for** iteration $q \subset Q$ in training **do**
  Sample $m$ instances $(X_g^k, Y_g) \sim \mathrm{P}_{\text{data}}(.)$
  Obtain $\boldsymbol{\theta_g}$ by optimizing $\tilde{\mathbf{G}}$ via Equation 2 with gradient descent
  Generate $\mathcal{S}_{\text{data}}$ via Equation 6
  **for** step $t \subset T$ in discriminator $\mathbf{D}$ **do**
    Obtain $\boldsymbol{\theta_d}$ by optimizing $\mathbf{D}$ via Equation 7 with gradient descent
**return** $\mathcal{S}_{data} \equiv \bar{X}_g^k, \bar{Y}_g$

**Algorithm 2:** Algorithm GANBLR−nAL

**Input** : $X_g^k, Y_g$
**Output** : Synthetic data $\bar{X}_g^k, \bar{Y}_g$

**for** iteration $q \subset Q$ in training **do**
  Sample $m$ instances $(X_g^k, Y_g) \sim \mathrm{P}_{\text{data}}(.)$
  Obtain $\boldsymbol{\theta_g}$ by optimizing $\tilde{\mathbf{G}}$ via Equation 2 with gradient descent
  Generate $\mathcal{S}_{\text{data}}$ via Equation 6
**return** $\mathcal{S}_{data} \equiv \bar{X}_g^k, \bar{Y}_g$

### D. GANBLR – No Adversarial Learning

It can be seen from Algorithm 1, that GANBLR can still fulfill its goal of synthesizing data without an adversarial learning component (i.e., $\mathbf{D}$). In practice, we can dump $\mathbf{D}$ from GANBLR – leading to a configuration that we call GANBLR−nAL. However, we argue that having an adversarial learning component can lead to much better data generation model as we discuss later in Section IV-D – where we compare GANBLR with GANBLR−nAL. Nonetheless, the GANBLR−nAL algorithm is provided in Algorithm 2:

### E. GANBLR – Summary

Let us briefly discuss two salient features of GANBLR. From Algorithm 1, it can be seen that GANBLR can generate the synthetic dataset $\mathcal{S}_{\text{data}}$ using Equation 6. As we mentioned in Section II, a desirable property of tabular generation models is generating the synthetic data with particular feature value (e.g., to resolve the imbalanced dataset limitation). Of course, the GANBLR's generator can simply sample the synthetic dataset by specifying the particular feature value with rejection sampling [21], and can be effective for imbalanced data generation.

## IV. Experiment and Analysis

Let us assess the efficiency of GANBLR in synthetic data generation in this section. We consider 15 commonly used datasets to compare GANBLR performance with three SOTA GAN models for tabular data generation. Specifically, we will evaluate the effectiveness of GANBLR in terms of:

**Machine learning utility** – which reflects the quality of the synthetic data.

**Statistical similarity** – which measures the statistical similarity between the synthetic and the real data.

**Interpretability** – which shows the interpretable capability of GANBLR.

Both the machine learning utility and the statistical similarity are standard measures to determine the quality of data generation algorithms [9]. Moreover, we perform various ablation studies to study a) the effect of GANBLR's hyper-parameter $k$ and b) to determine the effectiveness of adversarial component of GANBLR, by comparing GANBLR with GANBLR−nAL.

| Dataset | m | n | C | Size |
|---|---|---|---|---|
| Intrusion | 4M | 41 | 2 | Large |
| Pokerhand | 1M | 11 | 10 | Large |
| Covtype | 581012 | 55 | 7 | Large |
| Shuttle | 58000 | 9 | 7 | Large |
| Connect-4 | 67557 | 42 | 3 | Large |
| Credit | 50000 | 31 | 2 | Medium |
| Adult | 50000 | 14 | 2 | Medium |
| Chess | 28056 | 6 | 7 | Medium |
| letter_rocog | 20000 | 16 | 26 | Medium |
| Magic | 19020 | 11 | 2 | Medium |
| Nursery | 12960 | 8 | 5 | Small |
| Sign | 6500 | 8 | 3 | Small |
| Satellite | 6435 | 36 | 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.

Table II: Description of datasets.

### A. Experiment setup

*1) Datasets:* We use 15 commonly used classification datasets, 13 from UCI dataset repository and 2 from Kaggle — Credit and Loan. All these datasets have a specific dependent variable and a set of independent features. Among them, 5 are large datasets with more than 50K instances (denoted as Large), 5 are medium with less than 50K but greater than 15K instances (denoted as Medium), while the other 5 with less than 15K instances (denoted as Small). Their details are summarized in Table II.

*2) Baselines and Evaluation Metric:* We compare GANBLR with CTGAN, TableGAN and MedGAN. All baseline methods are trained with 150 epochs for 5 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 II that most datasets have $> 2$ classes, and hence, we have reported the accuracy (instead of widely used auc measure).

*3) Configuration and Running Environment:* The parameter $k$ in the experiments is set to 0 – that is the Bayesian network in the generator model is naive Bayes and in discriminator, it is Logistic Regression. In Section IV-D, we will study the effect of varying the value of $k$. GANBLR is coded in Python

| Dataset | Model | GANBLR | CTGAN | TableGAN | MedGAN |
|---|---|---|---|---|---|
| Intrusion | LR | **0.9316** | 0.9232 | 0.792 | 0.826 |
| | MLP | 0.901 | **0.926** | 0.739 | 0.803 |
| | RF | **0.9398** | 0.893 | 0.768 | 0.812 |
| | XGBT | **0.9318** | 0.918 | 0.721 | 0.831 |
| Pokerhand | LR | **0.533** | 0.469 | 0.471 | 0.371 |
| | MLP | **0.5518** | 0.435 | 0.39 | 0.382 |
| | RF | **0.572** | 0.502 | 0.41 | 0.39 |
| | XGBT | **0.5837** | 0.511 | 0.423 | 0.402 |
| Covtype | LR | 0.653 | **0.671** | 0.57 | 0.472 |
| | MLP | 0.673 | **0.6961** | 0.602 | 0.455 |
| | RF | 0.703 | **0.705** | 0.582 | 0.466 |
| | XGBT | **0.691** | 0.6827 | 0.591 | 0.497 |
| Shuttle | LR | **0.996** | 0.951 | 0.927 | 0.988 |
| | MLP | **0.993** | 0.972 | 0.912 | 0.982 |
| | RF | **0.9931** | 0.967 | 0.929 | 0.992 |
| | XGBT | **0.996** | 0.9801 | 0.92 | 0.991 |
| Connect | LR | **0.7517** | 0.702 | 0.663 | 0.692 |
| | MLP | **0.783** | 0.727 | 0.652 | 0.701 |
| | RF | **0.791** | 0.682 | 0.673 | 0.691 |
| | XGBT | **0.803** | 0.709 | 0.656 | 0.712 |
| Credit | LR | **0.9998** | 0.9979 | 0.9981 | 0.998 |
| | MLP | **0.9997** | 0.999 | 0.9972 | 0.9982 |
| | RF | **0.9996** | 0.9981 | 0.9987 | 0.999 |
| | XGBT | **0.9998** | 0.9976 | 0.998 | 0.998 |
| Adult | LR | 0.74 | **0.787** | 0.757 | 0.737 |
| | MLP | **0.842** | 0.831 | 0.761 | 0.739 |
| | RF | **0.81** | 0.792 | 0.783 | 0.721 |
| | XGBT | **0.851** | 0.839 | 0.775 | 0.733 |
| Chess | LR | **0.8478** | 0.693 | 0.722 | 0.577 |
| | MLP | **0.877** | 0.673 | 0.735 | 0.552 |
| | RF | **0.893** | 0.701 | 0.714 | 0.56 |
| | XGBT | **0.882** | 0.723 | 0.751 | 0.557 |
| Letter_recog | LR | **0.689** | 0.531 | 0.462 | 0.47 |
| | MLP | **0.739** | 0.601 | 0.392 | 0.462 |
| | RF | **0.713** | 0.566 | 0.452 | 0.47 |
| | XGBT | **0.736** | 0.539 | 0.478 | 0.483 |
| Magic | LR | **0.78** | 0.656 | 0.641 | 0.648 |
| | MLP | **0.789** | 0.672 | 0.632 | 0.668 |
| | RF | **0.803** | 0.698 | 0.665 | 0.671 |
| | XGBT | **0.812** | 0.681 | 0.632 | 0.676 |
| Nursery | LR | **0.8819** | 0.793 | 0.678 | 0.569 |
| | MLP | **0.902** | 0.839 | 0.681 | 0.575 |
| | RF | **0.919** | 0.802 | 0.676 | 0.582 |
| | XGBT | **0.93** | 0.836 | 0.688 | 0.585 |
| Sign | LR | **0.61** | 0.472 | 0.482 | 0.389 |
| | MLP | **0.649** | 0.479 | 0.473 | 0.401 |
| | RF | **0.61** | 0.4563 | 0.491 | 0.412 |
| | XGBT | **0.636** | 0.489 | 0.499 | 0.418 |
| Satellite | LR | **0.85** | 0.496 | 0.502 | 0.405 |
| | MLP | **0.887** | 0.502 | 0.492 | 0.388 |
| | RF | **0.879** | 0.51 | 0.511 | 0.392 |
| | XGBT | **0.89** | 0.499 | 0.502 | 0.41 |
| Loan | LR | **0.772** | 0.706 | 0.756 | 0.386 |
| | MLP | **0.801** | 0.683 | 0.762 | 0.392 |
| | RF | **0.816** | 0.731 | 0.757 | 0.406 |
| | XGBT | **0.822** | 0.739 | 0.759 | 0.403 |
| Car | LR | **0.85** | 0.713 | 0.62 | 0.611 |
| | MLP | **0.865** | 0.686 | 0.59 | 0.582 |
| | RF | **0.873** | 0.719 | 0.621 | 0.575 |
| | XGBT | **0.891** | 0.721 | 0.633 | 0.591 |
| Average | | **0.8151** | 0.7145 | 0.6651 | 0.6108 |

Table III: Machine Learning Utility on all datasets in TSTR

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

*4) Machine Learning Utility:* We will make use of the following two approaches to assess the machine learning utility performance:

**TSTR** – Training on Synthetic data, Testing on Real data.
**TRTR** – Training on Real data, and Testing on Real data.

To obtain the TSTR and TRTR performance of GANBLR and competing baseline methods, we will use four commonly used machine learning classification algorithms (Section IV-A2):

- Logistics Regression (LR),
- Multi-layer-Perceptron (MLP),
- Random Forest (RF), and
- Extreme Gradient Boosting Tree (XGBT).

For TSTR, we first split the real datasets into real training and real testing datasets: the real training datasets are used as the input for GANBLR and its baseline methods for training. Once training is completed, the synthetic datasets are generated. The synthetic training datasets are used to train the above-mentioned machine learning classification algorithms which will then be evaluated on the real testing datasets. The result of TSTR could not only show the realistic machine learning utility of all the compared methods, but also answer the question on "*Could synthetic data be used as substitute of the real data without significantly dropping the machine learning task performance?*". Ideally, higher the accuracy from TSTR (high machine learning utility), the better the data generation algorithm.

In contrast, TRTR is training the above-mentioned machine learning classification algorithms with real training datasets, and evaluating on real testing datasets. TRTR is included in the comparison for highlighting the ideal machine learning utility.

*5) Statistical Similarity:* Two metrics are used to quantitatively measure the statistical similarity between the real and the synthetic datasets generated by GANBLR and its baseline methods, that is:

- **Jensen-Shannon Divergence (JSD)**. The JSD quantifies the difference between the probability mass distribution of individual categorical feature in the real data and the synthetic dataset, and it is bounded between 0 and 1.
- **Wasserstein Distance (WD)**. Similarly, WD captures the earth moving distance on features between the real and synthetic dataset.

| Method | L˜Accuracy% | M˜Accuracy% | S˜Accuracy% |
|---|---|---|---|
| | TRTR | | |
| | 80.84% | 86.80% | 84.62% |
| | TSTR | | |
| GANBLR | **78.85%** | **84.02%** | **81.67%** |
| CTGAN | 75.27% | 74.88% | 64.37% |
| TableGAN | 66.95% | 71.72% | 60.87% |
| MedGAN | 67.28% | 68.58% | 47.36% |

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

## B. Results Analysis

*1) Results of Machine Learning Utility:* Table IV provides the averaged accuracy on Large, Medium and Small datasets. It is clear that GANBLR outperforms all other baseline methods in terms of accuracy based on TSTR . Particularly on the small and medium datasets, GANBLR has significantly better performance than other baseline methods. It is interesting to see that GANBLR's TSTR performance is close to TRTR performance, while none of the other baseline methods have the TSTR performance close to TRTR. Same findings could be seen in Table III which provide detailed TSTR performance for

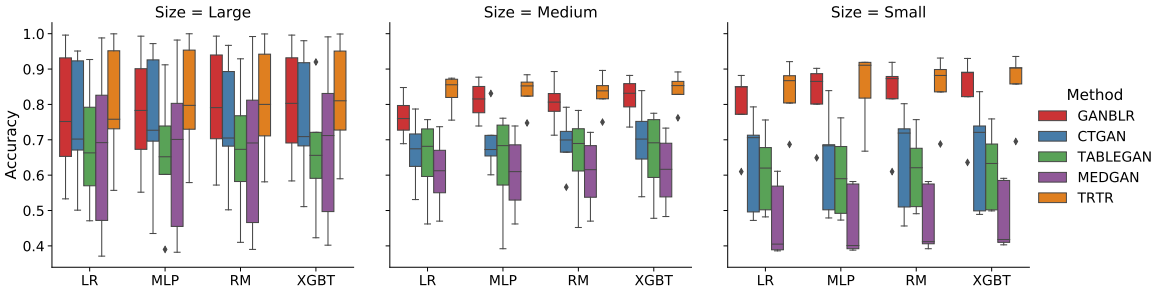



Figure 2: Machine Learning Utility comparison of GANBLR and competing baseline models on different sized datasets in terms of box-plot (Figure is seen best in color).

| Method | Large | | Medium | | Small | |
|---|---|---|---|---|---|---|
| | JSD | WD | JSD | WD | JSD | WD |
| GANBLR | 0.136 | **0.619** | 0.301 | 0.662 | **0.263** | **0.783** |
| CTGAN | **0.133** | 0.621 | **0.287** | 0.672 | 0.330 | 0.810 |
| TableGAN | 0.142 | 0.701 | 0.350 | **0.613** | 0.531 | 1.220 |
| MedGAN | 0.190 | 0.730 | 0.392 | 0.690 | 0.380 | 1.021 |

Table V: Comparison of Statistical Similarity measure of GANBLR with competing baseline models.

all datasets. These results are extremely encouraging, as they demonstrate that the synthetic data generated from GANBLR is far more useful for the machine learning tasks than from any other existing SOTA data generation algorithm.

While Figure IV provides averaged results, let us look at distribution of accuracies for different datasets. In Figure 2, we plot the box plots of the (TSTR) accuracy of GANBLR and its baseline methods on 4 machine learning algorithms. Again, we break the results in Large, Medium and Small datasets. We plot TRTR for sake of comparison as well. It can be seen that no matter the size of datasets, GANBLR significantly outperforms all the baseline methods. Especially, for Small and Medium datasets, the performance of GANBLR is extremely impressive as the box plots of GANBLR (blue) match highly to those of TRTR (purple).

*2) Statistical Similarity:* To obtain JSD results – for a dataset, each feature from synthetic dataset is measured against the same feature in 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 V – which as we discussed can be seen as the measure of statistical similarly between synthetic and original dataset.

It can be seen from Table V, that GANBLR stands-out when compared to the other baseline methods. If it is not the best, it is always the second best. Particularly, on Smaller datasets, GANBLR has smaller JSD and WD values than all the competing baselines, highlighting that it produces dataset of superiour quality. On Medium datasets, GANBLR has performance similar to CTGAN in terms of JSD and is the second best, while has WD performance similar to TableGAN and again is the second best. On Large datasets, GANBLR has the best performance in terms of WD, though it marginally loses to CTGAN in terms of JSD distance. Delving into why GANBLR has sub-optimal results on Medium datasets – we believe that this could be due to GANBLR's generator – Bayesian Network's ability to generate some feature value which are rarely seen in the real dataset. Clearly, statistical similarity evaluation based on JSD and WD does not credits the generation of data which is not present in real data, but is useful for classification task.

### C. Interpretation Analysis

Let us study the interpretable capability of our proposed model. We believe that a good interpretation in any tabular data generator should:

- provide the **local interpretability** with clear understanding of why a synthetic data point belongs to the generated synthetic label at any time during the training. E.g., given a generated synthetic data instance, the probability of each possible synthetic label should be provided.
- provide the **global interpretability** on how the features could impact the synthetic label generation generally. E.g., which feature has the largest impact on the synthetic label.

In order to investigate the interpretation capability of GANBLR, we use the CAR dataset for multi-class classification. Table VI shows the constrained weights from the generator in GANBLR. Here, one instance from CAR synthetic dataset is used for the local interpretability. It can be seen that the weights learned from GANBLR can be used to determine conditional probability i.e., $\mathrm{P}(Y|X)$. Therefore, during the training, GANBLR can easily explain locally on why the synthetic data feature from one instance belongs to the synthetic label.

In order to determine the credibility of the local interpretability of GANBLR, we employed the popular method – LIME, which could explain why features belongs to certain class on an instance level. Notably, we compare the weights of Table VI with the LIME experiment utilizing the XGBT algorithm (of course on the same data point) – Buying = 3, Maint = 2, Doors = 0, Persons = 1, Lug_boot = 2 Safety = 0, Class = 0. Note, in Table VI, the probability on each class is sorted in descending order with true class C = 0, achieving the highest probability (shown in red), followed by second highest probability on C = 2. It can be seen that in computing $\mathrm{P}(Y = 0|X)$, the feature Safety = 0 and Persons = 1 contribute the

most for this decision (probabilities shown in bold). Similarly, the results of LIME in Figure 3 from left to right are also sorted in descending order, and the sequence is exactly same as that of Table VI. Moreover, it is interesting to see exactly the same finding using LIME experiment, i.e., the highest probability is on C = 0, and the decision for this class is also based on Safety = 0 and Persons = 1 respectively. Of course, due to the limited space, we have presented just one demonstration of the interpretability comparison of GANBLR and LIME. However, we found similar pattern of coherence of GANBLR and LIME's interpretability, which indicates that the local interpretability of GANBLR is highly reliable even during the training phase and is equivalent to the LIME, which is basically interpretation of the model after the training.

In Figure 4, global interpretability of GANBLR at different training stage can be drawn by listing the weights learned from the generator in GANBLR. The darker color means bigger impact while the lighter color means smaller impact on the corresponding class. It can be seen that features impacts differently on different class labels during training phase, as can be seen at $epoch = 1$, $epoch = 50$ and $epoch = 100$. We can see that features Persons and Safety have the largest impact on the synthetic label 0, while features Maints, Persons and Safety have the largest impact on synthetic label 1, and features Persons, Lug_boot and Safety have the largest impact on synthetic label 2. However, feature Safety has far more impact on the synthetic label 3 than other 5 features, therefore, feature Safety is most important factor to decide the car with high value which is the meaning of C = 3. Again, the purpose of this analysis is to demonstrate GANBLR's global interpretable capability.

### *D. Ablation Analysis*

To illustrate the impact of hyper-parameter $k$ and the discriminative component on GANBLR, we conducted the ablation studies by changing the configuration of GANBLR as below:

**GANBLR-nAL** As we discussed in Section III-D, the GANBLR-nAL does not include the discriminator part and the generator has slightly simplified objective function – i.e., it is trained solely by maximizing $\log(\mathrm{P}(Y_g|X_g^k))$ as shown in Algorithm 2.

**k=0** In this experiment, the Bayesian Network generator in GANBLR has $k = 0$, i.e., we use a naive Bayes model, this means that no feature interaction is modelled.

**k=1** In this experiment, the generator in GANBLR is a Bayesian Network with $k = 1$, i.e., order 1 feature interactions are modelled.

**k=2** In this experiment, the generator in GANBLR is a Bayesian Network with $k = 2$, i.e., order 2 feature interaction are modelled.

We compare the performance of GANBLR and GANBLR-nAL using similar strategy that we used to compare GANBLR with other competing baselines. It can be seen from Table VII that GANBLR has better average performance than GANBLR-nAL especially on large datasets for various values of $k$. GANBLR is better than GANBLR-nAL except for $k = 1$ for medium and $k = 2$ for small. This highlights the significance of adversarial component in GANBLR formulation. Nonetheless, the comparison also highlights the usage of GANBLR-nAL as an effective sampling method which does not employ a game-theoretic adversarial learning.

Let us re-visit Table VII to see the impact of $k$. Well clearly, it can be seen that higher values of $k$ leads to better results for large and medium datasets. However, for small datasets, generally $k = 1$ has led to superiour performance. An obvious reason for this is that $k = 2$ might be over-fitting on the small datasets – and traditional bias-variance trade-off is coming into effect. Interestingly, this study revealed suitability of GANBLR's hyper-parameters to various sized datasets.

## V. Conclusion

In this work, we presented a novel technique to generate tabular data utilizing the GAN strategy. Our proposed GANBLR framework relies on discriminatively trained Bayesian Networks as the generator as well as discriminator, which learns by optimizing a game-theoretic objective function. We showed that GANBLR not only advances the existing SOTA GAN-based models but also leads to a framework with excellent interpretability during the training. We evaluated the data generation performance of GANBLR by comparing it against several SOTA baselines and analysed its performance in terms of machine learning utility as well as statistical similarity. The results show that the synthetic datasets generated from GANBLR have the best performance on machine learning utility and statistical similarity comparable to SOTA methods. The remarkable results of GANBLR demonstrate its potential for a wide range of applications which could greatly contribute to tabular data generation and augmentation in various sectors such as banking, insurance, health and many other industries. We highlight some future works as:

- We have constrained ourselves to $k \leq 2$ in this work. We are interested to see variation in the performance of GANBLR as the value of $k$ is increased.
- We have focused mainly on restricted BN model i.e., KDB models in our current GANBLR formulation – we are keen to study the model under un-restricted Bayesian Network models.
- Enhancing GANBLR to generate numerical attributes is also one direction, we are exploring.

## References

[1] B. Dai, S. Fidler, R. Urtasun, and D. Lin, "Towards diverse and natural image descriptions via a conditional gan," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2970–2979.

[2] 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.

[3] 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.

[4] 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.

| Class (C) | Buying (B=3) | Maint (M=2) | Doors (D=0) | Persons (P=1) | Lug_boot (L=2) | Safety (S=0) |
|---|---|---|---|---|---|---|
| P(C=0\|B, M, D, P, L, S) = **0.4892** | P(C=0\|B=3) = 0.131 | P(C=0\|M=2) = 0.352 | P(C=0\|D=0)=0.186 | P(C=0\|P=1)=**0.460** | P(C=0\|L=2)=0.205 | P(C=0\|S=0)=**0.572** |
| P(C=2\|B, M, D, P, L, S)=0.4211 | P(C=2\|B=3) = **0.288** | P(C=2\|M=2) = 0.202 | P(C=2\|D=0)=0.246 | P(C=2\|P=1)=0.154 | P(C=2\|L=2)=**0.307** | P(C=2\|S=0)=0.200 |
| P(C=1\|B, M, D, P, L, S)=0.0462 | P(C=1\|B=3) = 0.099 | P(C=1\|M=2) = 0.242 | P(C=1\|D=0)=0.241 | P(C=1\|P=1)=0.340 | P(C=1\|L=2)=0.322 | P(C=1\|S=0)=0.374 |
| P(C=3\|B, M, D, P, L, S)=0.0436 | P(C=3\|B=3) = 0.124 | P(C=3\|M=2) = 0.266 | P(C=3\|D=0)=0.172 | P(C=3\|P=1)=0.405 | P(C=3\|L=2)=0.165 | P(C=3\|S=0)=0.682 |

Table VI: Interpretation of GANBLR's generator weights at $epoch = 50$ on one instance of car dataset.

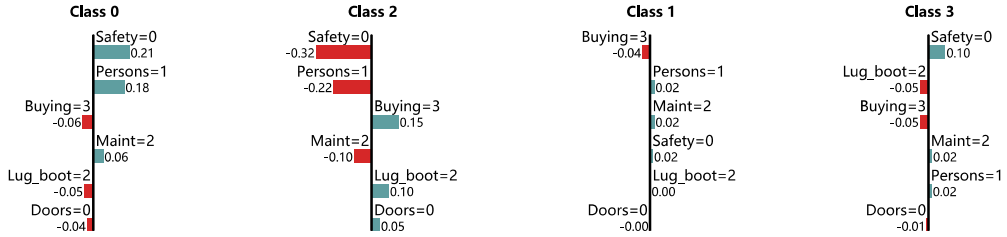
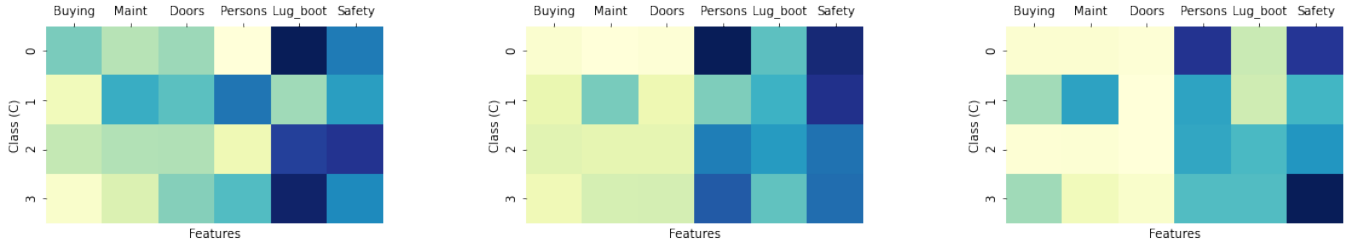



Figure 3: LIME explanation for synthetic data after training using XGBT*

* Probability of each class value are $P(C = 0) = 0.76$, $P(C = 2) = 0.22$, $P(C = 1) = 0.02$, $P(C = 3) = 0.01$




(a) $epoch = 1$, training accuracy $= 0.26$ (b) $epoch = 50$, training accuracy $= 0.76$ (c) $epoch = 100$, training accuracy $= 0.85$

Figure 4: Overall feature impact during the GANBLR training

| Hyper-parameter $k$ | Components | L˜TSTR | M˜TSTR | S˜TSTR |
|---|---|---|---|---|
| $k = 0$ | GANBLR | **78.85%** | 84.02% | **81.67%** |
| | GANBLR-nAL | 77.80% | **85.71%** | 80.21% |
| $k = 1$ | GANBLR | **82.91%** | **88.96%** | **85.63%** |
| | GANBLR-nAL | 82.07% | 87.37% | 84.70% |
| $k = 2$ | GANBLR | **85.75%** | **90.32%** | 80.75% |
| | GANBLR-nAL | 84.39% | 88.62% | **81.78%** |

Table VII: Ablation Analysis of GANBLR.


[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 7–10.

[11] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "xdeepfm: Combining explicit and implicit feature interactions for recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1754–1763.

[12] 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.

[13] N. A. Zaidi, M. J. Carman, J. Cerquides, and G. I. Webb, "Naive-bayes inspired effective pre-conditioner for speeding-up logistic regression," in *2014 IEEE International Conference on Data Mining*, 2014, pp. 1097–1102.

[14] E. Fetaya, J.-H. Jacobsen, W. Grathwohl, and R. Zemel, "Understanding the limitations of conditional generative models," 2020.

[15] N. A. Zaidi, Y. Du, and G. I. Webb, "On the effectiveness of discretizing quantitative attributes in linear classifiers," *IEEE Access*, vol. 8, pp. 198 856–198 871, 2020.

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

[17] 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.

[18] Y. Yu, B. Tang, R. Lin, S. Han, T. Tang, and M. Chen, "Cwgan: Conditional wasserstein generative adversarial nets for fault data generation," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 2713–2718.

[19] T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri, "On discriminative Bayesian network classifiers and logistic regression," *Machine Learning*, vol. 59, no. 3, pp. 267–296, 2005.

[20] H. Guo and W. Hsu, "A survey of algorithms for real-time bayesian network inference," in *Join Workshop on Real Time Decision Support and Diagnosis Systems*, 2002.

[21] A. Grover, R. Gummadi, M. Lazaro-Gredilla, D. Schuurmans, and S. Ermon, "Variational rejection sampling," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2018, pp. 823–832.