FakeTables: Using GANs to Generate Functional Dependency Preserving Tables with Bounded Real Data

Haipeng Chen[1*], **Sushil Jajodia**[2], **Jing Liu**[2], **Noseong Park**[2*], **Vadim Sokolov**[2] and **V. S. Subrahmanian**[1]

[1]Dartmouth College

[2]George Mason University

haipeng.chen@dartmouth.edu, {jajodia,jliu30,npark9,vsokolov}@gmu.edu, vs@dartmouth.edu

Abstract

In many cases, an organization wishes to release some data, but is restricted in the amount of data to be released due to legal, privacy and other concerns. For instance, the US Census Bureau releases only 1% of its table of records every year, along with statistics about the entire table. However, the machine learning (ML) models trained on the released sub-table are usually sub-optimal. In this paper, our goal is to find a way to augment the sub-table by generating a synthetic table from the released sub-table, under the constraints that the generated synthetic table (i) has similar statistics as the entire table, and (ii) preserves the functional dependencies of the released sub-table. We propose a novel generative adversarial network framework called ITS-GAN, where both the generator and the discriminator are specifically designed to satisfy these two constraints. By evaluating the augmentation performance of ITS-GAN on two representative datasets, the US Census Bureau data and US Bureau of Transportation Statistics (BTS) data, we show that ITS-GAN yields high quality classification results, and significantly outperforms various state-of-the-art data augmentation approaches.

1 Introduction

Over the past few decades, machine learning related research has seen remarkable developments, driven in part by the tremendous amount of data that has become available. In many cases, an organization such as a government or company wants to release some data to the public, but is restricted in the amount of data released due to legal, privacy and other concerns. For instance, the U.S. Census Bureau releases 1% of its records - but these do not completely represent the US population. At the same time, averages about the entire population's income and some other attributes are also released. Many U.S. government survey datasets are released in such a manner and this has been the practice in various fields, such as business, operations research and social science.

Data augmentation has recently drawn considerable attention in the machine learning (ML) community, where the objective is to improve ML models by enhancing the quality of data. In this paper, we study an *incomplete table synthesis* (ITS) problem for tabular data augmentation, where we wish to augment the released incomplete sub-table of records X' by synthesizing a new table Y of records, so that a machine learning model trained on the augmented table $X' \cup Y$ works better for the full table X (which X' originated from) than a model that is trained solely on X'.

GANs have been widely adopted in image generation [Isola *et al.*, 2017; Zhu *et al.*, 2017], language analytics, [Pascual *et al.*, 2017] — there also has been some very recent interest w.r.t. tables [Park *et al.*, 2018]). However, using GANs to solve the ITS problem requires addressing two new challenges. First, while we wish to learn a generative model from the sub-table X', we also need to maintain the broad statistics of the table X. Second, relational tables usually have an associated set of functional dependencies (FDs) — see Definition 3.1. As FDs capture dependencies between attributes, it is critical to ensure that $X' \cup Y$ also satisfy these dependencies, otherwise the integrity of $X' \cup Y$ is questionable. To the best of our knowledge, our paper is the first to solve the problem of tabular data synthesis with these two constraints.

To tackle the two challenges, we propose a novel framework called ITS-GAN, which makes three innovations in customizing traditional GANs to efficiently learn a synthetic table generator. First, we add two extra loss terms into the original generator loss function to encode these two constraints in the GAN training process. A major difficulty of encoding the FD constraints into the loss function is that the FDs associated with a table are usually expressed at the *schema (column) level*. For example, in Table 1, an example *schema-level* FD is "Position" determines "Salary". One schema-level FD generates numerous *record-level* FDs (see Definition 3.2). An example of a record-level FD in Table 1 is, the Position "CEO" determines the Salary "$2M". While we can enumerate the two record-level FDs in Table 1, it is impossible to explicitly list all the record-level FDs in real-world large datasets with numerous records. Therefore, our second innovation is to train an autoencoder (one for each schema-level FD) to model the record-level FDs. The autoencoders are pre-trained to reproduce the FDs and then fed into the

*Co-corresponding authors.

generator loss function. Moreover, to improve the training of the GAN discriminator, we make our third innovation via a new customized calibration of the discriminator's input using the FD errors calculated by the pre-trained autoencoders.

We provide a theoretical analysis of ITS-GAN under certain conditions as well as extensive empirical evaluations on two widely used datasets - Census and BTS. The empirical results show that ITS-GAN significantly improves the performance of classification tasks compared with state-of-the-art data augmentation approaches. An ablation study also demonstrates the efficacy of ITS-GAN by taking into account the table statistics and FD constraints in the ITS problem.

2 Related Work

In this section, we describe related work. We discuss classical augmentation methods, followed by recent advanced GAN-based methods.

2.1 Classical Augmentation Methods

Various data augmentation methods are used to handle imbalanced data — SMOTE [Chawla *et al.*, 2002] is among the most popular approaches. SMOTE generates weighted combinations of randomly selected existing records. Borderline SMOTE (BSMOTE) [Han *et al.*, 2005] uses a similar method around a class boundary. ADASYN [He *et al.*, 2008] was designed on top of SMOTE, where weights are adjusted in order to account for various factors.

The condensation method [Aggarwal and Yu, 2004] is a classical non-machine learning method to synthesize tables, which assumes that a column follows a specific statistical distribution and synthesizes values for the column by exploiting its assumed statistical characteristics. In many cases, however, its synthesis range is narrow and does not reproduce the entire value range of the column.

All these approaches are not suitable for our problem because they do not support the preservation of FDs.

2.2 GAN-based Augmentation Methods

GANs consist of two neural networks: a generator and a discriminator [Goodfellow *et al.*, 2014], which are alternately updated as a two-player zero-sum minimax game:

$$\min_G \max_D V(G,D) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z}[\log(1-D(G(\mathbf{z})))], \quad (1)$$

where $V(G,D)$ is the value function of the discriminator, p_{data} is the probability distribution of the real dataset, $\mathbf{z}$ is the input noise of the generator which follows the prior distribution p_z, $G(\mathbf{z})$ is a generator function mapping $\mathbf{z}$ to the generated data $\mathbf{x}$, and $D(\mathbf{x})$ is a discriminator function which specifies the probability that a discriminator input $\mathbf{x}$ is from the real dataset.

GANs have been exploited to synthesize electronic health records in [Choi *et al.*, 2017; Baowaly *et al.*, 2018]. However, they are mostly interested in synthesizing discrete values in patient records and cannot be directly applied to synthesize tables with other types of values. More recently, TableGAN was proposed to generate fake tables using GANs given an original table [Park *et al.*, 2018]. However, TableGAN synthesizes a full table X directly and does not consider FDs. DCGAN [Radford *et al.*, 2015] is a benchmark image generation framework which can be customized to synthesize tables [Park *et al.*, 2018], but it also fails to consider the table statistics and the FD constraints.

Position	Team	Name	Salary
CEO	Null	John Doe	$2M
Manager	AI Team	Jane Doe	$1M
Manager	DB Team	John Smith	$1M

Table 1: An example of payroll table.

3 Problem Description

Let X denote a *full table* of records. We use X_i to denote the i'th record and X^j to denote the j'th column of X. A *sub-table* $X' \subseteq X$ is a subset of rows of X. Table 1 shows an example of a full table X, where the first row X_1 = [CEO, Null, John Doe, $2M], the first column X^1 = [CEO, Manager, Manager], and the first two rows form a sub-table X'.

Each full table is accompanied by a vector $\bar{X}$ specifying the average value of each numerical column. For example, the average value of X^4 is 1.33M. For relational tables, there usually exist a set of schema-level functional dependencies that preserve the validity and consistency of the tabular data.

Definition 3.1. *Suppose A, B are two disjoint subsets of the schema (columns) of X'. A **schema-level functional dependency (FD)** F associated with X' has the form*

$$F : A \rightarrow B.$$

This says that for any record (row), the column values in A uniquely determine the column values in B.

The set of all schema-level FDs is denoted as $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$. E.g., in Table 1, 'Position' uniquely determines 'Salary', while 'Position' and 'Team' uniquely determines 'Name'. Therefore, the full set of FDs in Table 1 is $\mathcal{F} = \{F_1, F_2\}$, where: $F_1 : X^1 \rightarrow X^4$, and $F_2 : X^1, X^2 \rightarrow X^3$. As opposed to the schema-level FDs, we also define:

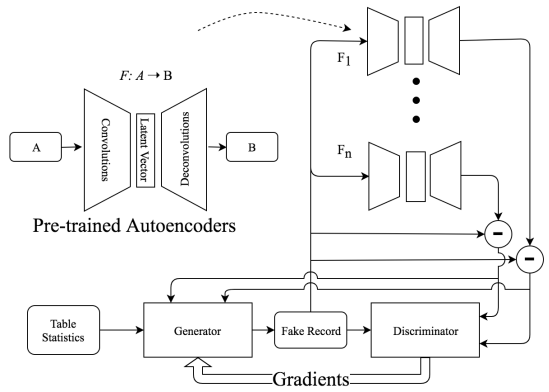
Definition 3.2. *Let $a \in A, b \in B$ denote two entries in A and B, a **record-level** FD f has the form*

$$f : a \rightarrow b.$$

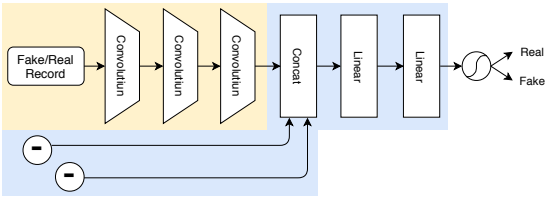
It says that the specific value of 'a' uniquely determines the specific value of 'b'.

We refer to the example in the Introduction to illustrate the difference between a schema-level and a record-level FDs. As introduced earlier, there are many real-world scenarios where an organization wants to release some data to the public, but is restricted in the amount of data. The common practice of these organizations is to release a small portion of the dataset, along with some statistics of the entire dataset. To improve the ML model trained on the released dataset, we solve the following ITS problem.

Definition 3.3. ***Incomplete Table Synthesis (ITS)**: Given a sub-table $X' \subseteq X$, a column-wise average $\bar{X}$ of the numerical attributes of the full table X and a set of FDs $\mathcal{F}$ of the*



(a) Overall architecture



(b) Discriminator architecture. The circles with '-' are the inputs from the autoencoders.

Figure 1: The architectures of ITS-GAN.

sub-table X', our objective is to generate a synthetic table Y such that

- *Y has the same number of columns as X and X'.*
- *Y satisfies the set of FDs in $\mathcal{F}$.*
- *Y maintains the column-wise average $\bar{X}$ of X.*

4 ITS-GAN

Adapting classical GANs to address the ITS problem poses two challenges: (i) We need to learn a generative model from the sub-table X' and maintain the column-wise statistics associated with the full table X. These two objectives contradict each other when $\bar{X}' \neq \bar{X}$. (ii) During the table synthesis process, we need to maintain the record-level FDs.

To handle the challenges, we propose the ITS-GAN (Incomplete Table Synthesis with Generative Adversarial Network) framework shown in Figure 1. ITS-GAN has three novelties. First, it trains one autoencoder-based model for each schema-level FD in $\mathcal{F}$. Second, it incorporates the table statistics and FD constraints in training the generator so that the generator establishes a trade-off between the two contradicting objectives mentioned above. Third, it calibrates the discriminator using the pre-trained autoencoders, so that the learned discriminator is able to capture the FD constraints.

4.1 Modelling FDs with Autoencoders

Autoencoders were initially proposed to capture the latent vector of an input in between an encoder and a decoder [Hinton and Salakhutdinov, 2006; Baldi, 2011]. In ITS-GAN, we train an autoencoder to reproduce the right-hand side of a record-level functional dependency, given its left-hand side. There is one autoencoder for each schema-level functional dependency and it is pre-trained with all record-level instances in the sub-table X'.

Recall that a schema-level functional dependency is represented as $F : A \rightarrow B$, where A and B are column-wise subsets of the original table. As a result, the inputs of an autoencoder are the entries of A, and the outputs are the corresponding entries in B. As shown in Figure 1 (a), for each autoencoder, its encoder component uses a convolution layer with a filter size of 5 and a stride of 2, while the decoder component uses a transpose convolution layer with a filter size of 5 and the same stride value.

4.2 Generator

To handle the functional dependency and table statistics constraints, we propose the following adapted loss function for the generator:

$$\mathcal{L}'_G = \mathcal{L}_G + \alpha \|\bar{Y} - \bar{X}\|_1 + \beta \sum_i \sum_j \|B'_{ij} - B_{ij}\|_1, \quad (2)$$

where $\mathcal{L}_G$ is the original loss function in Eq. (1). $\|\bar{Y} - \bar{X}\|_1$ is an error term which penalizes the difference between the column-wise average of the generated table Y and the original table X. Throughout this paper, we use L1-norm as the distance measure between vectors. $\sum_i \sum_j \|B'_{ij} - B_{ij}\|_1$ is another error term which characterizes the set of FD constraints, where B'_{ij} is the right hand side of the i^{th} schema-level FD in the j^{th} record instance produced by the generator, and B_{ij} is the corresponding autoencoder's anticipated right hand side of the i^{th} schema-level FD (given the left hand side in the generated j^{th} record). α and β are hyper-parameters which indicate the weights of different error terms.

Approximating statistics Error. Note that in mini-batch training (e.g., a batch size of 64), an average of one mini-batch does not accurately represent the statistics that the generator learns, thus providing a biased estimation of the table statistics error $\|\bar{Y} - \bar{X}\|_1$. To get a better approximation, we update the average value using a moving average rule [Winters, 1960], where the moving average $\hat{\bar{Y}}_{t+1}$ of the $(t+1)^{th}$ epoch is a weighted combination of the moving average of the t^{th} epoch and the average of the $(t+1)^{th}$ epoch:

$$\hat{\bar{Y}}_{t+1} = (1-w)\hat{\bar{Y}}_t + w\bar{Y}_{t+1}. \quad (3)$$

Here w is a hyper-parameter indicating the update rate — we set $w = 0.01$ in our experiments. In our generator, we use 3 layers of transpose convolutions, where the filter size is 5 and the stride value is 1.

4.3 Discriminator

The architecture of our discriminator is shown in Figure 1 (b). The process has two stages highlighted in yellow and blue respectively. The input (i.e., either fake records from the generator or the real data) is first filtered with three convolution layers in the first stage and later concatenated with the FD error terms from the autoencoders in the second stage. In each convolution layer, we use a convolution with the same filter size and stride value as that of the encoder.

Note that given a real record (row), FD error vectors will contain values close to zero if autoencoders are properly pre-trained. As a consequence, it will be relatively easy to detect a fake record if the norm of the error vector of the fake record is large, rather than small. This gives the generator a strong incentive to ensure that the FD error rate is small. Recall that we impose this requirement in the definition of the loss function of the generator (cf. Eq. (2)). Because of this, the ITS-GAN architecture is optimized toward minimizing violations of FDs through the FD error terms. To capture this property and improve the training of both the discriminator and the generator, we concatenate the output of the convolutional layers with the functional dependency errors (obtained from the pre-trained autoencoder). More specifically, for the j^{th} fake record produced by the generator, we concatenate it with a set of error terms $\|B'_{ij} - B_{ij}\|$ for all $F_i \in \mathcal{F}$. After concatenation, two fully connected layers follow, and a Sigmoid activation is used to generate the prediction for the record (i.e., real or fake). We train the discriminator with the same original loss function as in Eq. (1) — another option is the loss in [Gulrajani *et al.*, 2017] but we found that the original loss sometimes produces better results in our task.

5 Theoretical Results

As described earlier, we introduce two new error terms on the generator loss function to handle the table statistics and functional dependency constraints. However, if the deviation from X' for the two error terms is huge for a given record, it will be easy for the discriminator to classify that record as fake. Thus, the generator should find a balance in which the match between ground-truth statistics and $X' \cup Y$ statistics is good, but at the same time, the discriminator cannot distinguish between real and fake records. The following theorem describes the equilibrium solution to the minimax game between the generator and the discriminator when there is no penalty for errors in the statistics of $X' \cup Y$ compared to X.

Theorem 5.1. *The equilibrium distribution D^* of the discriminator and p^*_G of the generator in ITS-GAN is, $D^* = \frac{p_{X'}}{p_{X'}+p^*_G}$, and $p^*_G = p_{X'}$, if there is no table statistics error term (i.e., the second term on the right hand side of Eq.* (2)*) in the generator loss function of ITS-GAN.*

Proof. We denote a single record in a table as a vector $\mathbf{x}$. For a fixed generator G, the discriminator's objective is to minimize the following loss function:

$$\mathcal{L}_D = \int_{\mathbf{x}} -p_{X'}(\mathbf{x}) \log(D(\mathbf{x})) - p_G(\mathbf{x}) \log(1 - D(\mathbf{x}))d\mathbf{x}.$$

Performing a point-wise (i.e., for each $\mathbf{x} \sim p_{X'}$) optimization of the objective, we can obtain the *best response* of the discriminator as

$$D^*(\mathbf{x}) = \frac{p_{X'}(\mathbf{x})}{p_{X'}(\mathbf{x}) + p_G(\mathbf{x})}.$$

Substitute it into the loss function of the generator, we have

$$\begin{aligned}\mathcal{L}_G(D^*) =& E_{\mathbf{x}\sim p_{X'}}\big[\log \frac{p_{X'}(\mathbf{x})}{p_{X'}(\mathbf{x}) + p_G(\mathbf{x})}\big] + \\ & E_{\mathbf{x}\sim p_G}\big[\log \frac{p_G(\mathbf{x})}{p_{X'}(\mathbf{x}) + p_G(\mathbf{x})}\big] + \beta \sum_i \sum_j \|B'_{ij} - B_{ij}\|_1\end{aligned}$$

Note that $E_{\mathbf{x}\sim p_{X'}}[-\log 2] + E_{\mathbf{x}\sim p_g}[-\log 2] = -\log 4$. Adding the above two equations, and following the definition of KL-divergence, we have

$$\begin{aligned}\mathcal{L}_G(D^*) =& -\log 4 + KL\big(p_{X'}\big|\frac{p_{X'} + p_G}{2}\big) + \\ & KL\big(p_G\big|\frac{p_{X'} + p_G}{2}\big) + \beta \sum_i \sum_j \|B'_{ij} - B_{ij}\|_1 \\ =& -\log 4 + 2JS(p_{X'}||p_G) + \beta \sum_i \sum_j \|B'_{ij} - B_{ij}\|_1,\end{aligned}$$

where the second equality uses the definition of JS-divergence. Since the JS-divergence between two distributions is always non-negative and zero if they are equal, as a result, we see that the minimum of the second term on the right hand side is obtained when $p^*_G = p_{X'}$.

We now look at the third term, which is an indicator of the functional dependency error. For this error term, the optimum is obtained when $B'_{ij} = B_{ij}$, for each record j and each functional dependency $F_i \in \mathcal{F}$. This implies that, the data distribution of a subset of columns in the generated table Y should be the same as that of the sub-table X'. As a result, the condition $p^*_G = p_{X'}$ does not violate the necessary condition of obtaining optimum of the functional dependency error. Therefore, under equilibrium, we have $p^*_G = p_{X'}$, and $D^* = \frac{p_{X'}}{p_{X'}+p^*_G}$.

□

Note that $p^*_G = p_{X'}$ is not a sufficient condition for the optimality of minimizing the functional dependency loss, and its optimum is obtained (with a more strict condition) by enforcing the modelled functional dependency function (via the autoencoders). Moreover, the equilibrium obtained in practice is also dependent on how well the FDs are fitted by the autoencoders. The following two corollaries can be immediately obtained from Theorem 5.1:

Corollary 5.1.1. *The equilibrium D^* and p^*_G described in Theorem 5.1 can still obtained when $\bar{X}' = \bar{X}$ (i.e., the column-wise average values of the full table and the sub-table are the same) and any if one of the following conditions hold:*

- *Full batch gradient descent is performed in training generator neural networks.*
- *The moving average vector update weight $w \to 0$ or the weight $\alpha \to 0$ in the generator loss function with enough training epochs.*

Corollary 5.1.2. *When $\bar{X}' \neq \bar{X}$, for discriminator, we still have $D^* = \frac{p_{X'}}{p_{X'}+p^*_G}$, while for generator, $p^*_G(\mathbf{x})$ is biased towards $\bar{X}$ compared with $p_{X'}$.*

Corollary 5.1.2 implies that our synthesized new table Y of records is able to find a fine balance point which trades off among the three conditions specified in Definition 3.3.

6 Empirical Results

We describe detailed experiment settings and results on two representative datasets with various evaluation metrics.

6.1 Datasets and Experiment Settings

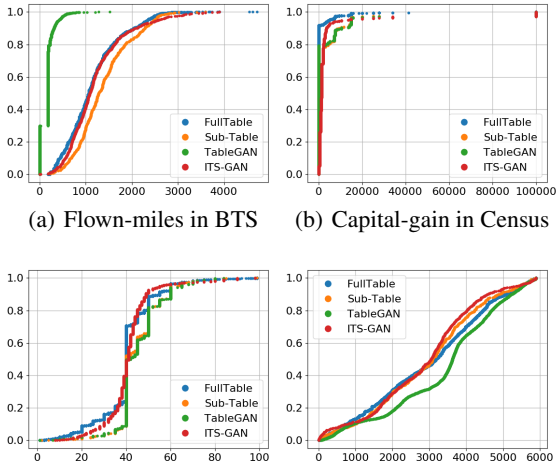
Datasets. The ITS problem occurs frequently in real world scenarios. We select the following two representative datasets to evaluate the performance of ITS-GAN. (i) **Census** [Kohavi, 1996] is a dataset that contains 30K samples of population data released by the US Censors Bureau. (ii) **BTS**[1] is the Air Carrier Statistics (ACS) dataset from the US Bureau of Transportation Statistics that describes 80K domestic air ticket sales in the US in a month. In particular, these two datasets are considered as standards in many fields such as Social Science, Civil Engineering, Transportation, etc.

Preprocessing. ITS-GAN synthesizes three types of columns. (i) Continuous Nnumeric: It is straightforward to learn and generate real numbers. (ii) Discrete numeric: After generating a numeric value v, we round it to the nearest column value in X', e.g., 1.5 will be rounded to 3 if the corresponding column in X' only contains $\{-10, 3, 4, 5\}$. (iii) Categorical: We use a label encoding method[2] to assign an integer value to each category or class, after which generating categorical values is equivalent to the discrete numeric case. For both Census and BTS datasets, the ground-truth full-table statistics (in addition to the released samples) are also released. However, we do not know their original full tables as these are not publicly available. Hence, we cannot compare our synthetic tables with the unknown full tables. Therefore, we create our own evaluation data from their released data as follows. We treat the Census and BTS data we have as the full table (i.e. X) and then create sub-tables (X') consisting of $p\%$ of X. We test $p \in \{1, 5\}$ as we are interested in very challenging situations where p is very small. Each of the two datasets has one functional dependency: gender, relationship determine marital-status in Census and source/destination airports, flown distance determine overall distance in BTS.

Experiment Settings. Our experiments are run on Ubuntu 18.04 with CUDA 10, Tensorflow r12. The batch size is 64, with learning rate of 0.0002 and Adam optimizer (with beta parameter of 0.5). The generator and discriminator are trained alternately as in [Goodfellow *et al.*, 2014]. For the weights in Eq. (2), we use $\alpha = \beta = 3$ for Census and 5 for BTS.

6.2 CDF Comparison

To evaluate whether a generated table Y is close to the original full table X, we first compare the cumulative distribution functions (CDFs) of X and Y. We compare with the state-of-the-art table synthesis approach, TableGAN [Park *et al.*, 2018]. Figure 2 displays the CDFs of some selected schema (columns). In many cases, CDFs of ITS-GAN in red are close to CDFs of full table X in blue, suggesting that ITS-GANperforms very well. In Figure 2 (d), however, ITS-GAN is close to sub-table X' in orange around the X-axis value range of 3000-5000, which means our generation does not always successfully capture the statistics of the full table. In 70% (resp. 60%) of the columns, ITS-GAN is closer to full table than to sub-table in Census (resp. BTS) in terms of the Earth mover distance [Pele and Werman, 2009]. In contrast, the CDFs of TableGAN in green are mostly close to CDFs of sub-table X' in orange, which demonstrates that ITS-GAN is superior to TableGAN in terms of synthesizing a full table.



(a) Flown-miles in BTS (b) Capital-gain in Census

(c) Hours-per-week in Census (d) Airport-group in BTS

Figure 2: Cumulative distributions of some selected columns.

6.3 Table Augmentation Performance Comparison

To test the utility of augmentation using the synthetic table generated by ITS-GAN, we run classification tasks on the two datasets. More specifically, we predict the "hours-per-week" in the Census data and "ticket price" in the BTS data. The labels are set as 1 (resp. 0) if the value is larger or equal to (resp. smaller than) the median. To find the classifier with the best performance, we do a grid search over AdaBoost, RandomForest, GradientBoosting with various hyper-parameters and using F-1 as the model selection criterion.

Baseline Methods. The set of baseline methods include:

- CM [Aggarwal and Yu, 2004], ROS [Lemaître *et al.*, 2017], ADASYN [He *et al.*, 2008], SMOTE [Chawla *et al.*, 2002] and its variants BSMOTE [Han *et al.*, 2005], SVMSMOTE [Demidova and Klyueva, 2017], SMOTENC [Chawla *et al.*, 2002], etc.
- DCGAN [Radford *et al.*, 2015], which is a benchmark GAN framework for generating images. DCGAN is customized to generate tabular data.
- TableGAN [Park *et al.*, 2018], which is the state-of-the-art GAN-based table synthesis approach synthesizes a full table and does not consider FDs.

The hyper-parameters are set as stated in the associated papers (when specified), or otherwise tuned via a grid search.

Table 2 summarizes the classification results. Note that some baselines do not properly perform the augmentation task – in particular, SMOTENC and ADASYN produce runtime errors in some sub-tables. This happens when the sub-table size is small and/or biased and their subroutines cannot process it. SMOTE and BSMOTE are able to augment

[1] https://www.transtats.bts.gov/Tables.asp?DB_ID=111

[2] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

	$\|X'\|$	Method	F-1	Recall	Precision
Census	15% of X		0.463	0.386	0.581
	5% of X	X' only	0.452	0.664	0.360
		$X'\cup$ SMOTE	0.466	0.749	0.348
		$X'\cup$ BSMOTE	0.467	**0.794**	0.339
		$X'\cup$ SVMSMOTE	0.436	0.610	0.366
		$X'\cup$ SMOTEEN	0.427	0.560	0.363
		$X'\cup$ ROS	0.451	0.631	0.374
		$X'\cup$ DCGAN	0.415	0.509	0.353
		$X'\cup$ TableGAN	0.435	0.579	0.354
		$X'\cup$ ITS-GAN (no Statistics)	0.458	0.560	**0.415**
		$X'\cup$ ITS-GAN (no FDs)	0.446	0.521	0.393
		$X'\cup$ ITS-GAN	**0.478**	0.678	0.382
	1% of X	X' only	0.438	0.649	0.353
		$X'\cup$ SMOTE	0.438	0.677	0.342
		$X'\cup$ BSMOTE	0.439	0.678	0.341
		$X'\cup$ SVMSMOTE	0.415	0.564	0.343
		$X'\cup$ SMOTEEN	0.399	0.551	0.322
		$X'\cup$ ROS	0.439	0.678	0.341
		$X'\cup$ DCGAN	0.405	0.503	0.349
		$X'\cup$ TableGAN	0.418	0.610	0.336
		$X'\cup$ ITS-GAN (no Statistics)	0.438	0.821	0.322
		$X'\cup$ ITS-GAN (no FDs)	0.443	**0.825**	0.323
		$X'\cup$ ITS-GAN	**0.462**	0.663	**0.364**
BTS	15% of X		0.652	0.953	0.505
	5% of X	X' only	0.648	0.773	0.560
		$X'\cup$ SMOTE	0.635	0.773	0.560
		$X'\cup$ BSMOTE	0.616	0.700	0.560
		$X'\cup$ SVMSMOTE	0.576	0.585	0.600
		$X'\cup$ SMOTEEN	0.522	0.450	0.626
		$X'\cup$ ROS	0.547	0.537	0.584
		$X'\cup$ DCGAN	0.536	0.496	**0.629**
		$X'\cup$ TableGAN	0.602	0.698	0.552
		$X'\cup$ ITS-GAN (no Statistics)	0.636	0.836	0.538
		$X'\cup$ ITS-GAN (no FDs)	0.635	0.836	0.537
		$X'\cup$ ITS-GAN	**0.666**	**0.983**	0.503
	1% of X	X' only	0.640	0.777	0.554
		$X'\cup$ SMOTE	0.580	0.592	0.582
		$X'\cup$ BSMOTE	0.574	0.583	0.586
		$X'\cup$ SVMSMOTE	0.539	0.521	0.576
		$X'\cup$ SMOTEEN	0.531	0.502	0.570
		$X'\cup$ ROS	0.555	0.528	**0.592**
		$X'\cup$ DCGAN	0.492	0.449	0.546
		$X'\cup$ TableGAN	0.626	0.829	0.526
		$X'\cup$ ITS-GAN (no Statistics)	0.630	0.804	0.544
		$X'\cup$ ITS-GAN (no FDs)	0.621	0.808	0.530
		$X'\cup$ ITS-GAN	**0.648**	**0.892**	0.525

Table 2: Table augmentation performance comparison. We perform classification of the Hours-per-week label for the Census data and the Ticket price label in BTS data (class 1: $\geq$ median, class 0: $<$ median). Both comparisons against baselines and the ablation study are presented in the table. Results are averaged over 10 different X'.

all sub-tables in a reliable manner. DCGAN is always inferior to TableGAN as observed in [Park *et al.*, 2018] and we remove CM from the table due to its poor generation quality. ITS-GAN outperforms all the baseline methods in terms of F1 score and precision in Census and achieves the best F1 score and recall in BTS. For recall in Census and precision in BTS, Boundary-SMOTE (BSMOTE), which augments around classification boundary, shows the best performance. However, BSMOTE's overall F1 measure is inferior to ITS-GAN. The other baselines show relatively unreliable performance. Moreover, GAN-based augmentation methods show more reliable performance when X' is 1% of X. The performance gap between the 1% and 5% cases for the baseline methods is around 3-5% whereas it is around 1-2% for ITS-GAN and TableGAN. We also calculate the statistical significance between ITS-GAN and BSMOTE across all predictions — the p-value is 0.002.

To further show the significance of table augmentation on ML tasks, we also compare with the ML model trained on a larger dataset which contains the sub-table X'. This dataset contains 15% of records from the full table X. Surprisingly, our method shows comparable performance to the models trained on the much larger dataset, and performs even better in some cases, e.g., the F1 score of ITS-GAN is 0.478 in Census, and is 0.463 for the ML model trained on the 15% dataset.

Ablation Study. To check the efficacy of the table statistic information and FD constraints, we also do an ablation study. We consider two variations of ITS-GAN: ITS-GAN (no FDs) and ITS-GAN (no Statistics) where autoencoders or the statistics matching loss of the generation are missing respectively. ITS-GAN without both of these components is comparable to TableGAN, so we exclude it. The results are also shown in Table 2. We see that for all cases, ITS-GAN outperforms the two variants with one component missing. This demonstrates that ITS-GAN is able to exploit both the table statistic information and the FD constraints to better augment a sub-table.

7 Conclusion

We presented a tabular data augmentation framework called ITS-GAN. Our approach is widely applicable under scenarios where we are given a sub-table from a full table, along with some statistics associated with the full table. This problem setting occurs very frequently in many real-world cases, especially when an organization like the government is restricted in the amount of data that it can release, e.g. due to legal and privacy concerns. ITS-GAN adapts GANs to maintain both the statistics of the full table and the FDs in the released sub-table. We evaluated the performance of data augmentation for ITS-GAN on two very important datasets, Census and BTS. The results show that our approach is able to significantly improve the classification tasks, and outperforms the state-of-the-art tabular data augmentation approaches by a significant edge.

Acknowledgements

This work is supported by ONR grants N00014-18-1-2670 and N00014-16-1-2896 and ARO grant W911NF-13-1-0421. Authors are in alphabetical order.

References

[Aggarwal and Yu, 2004] Charu C. Aggarwal and Philip S. Yu. A condensation approach to privacy preserving data mining. In Elisa Bertino, Stavros Christodoulakis, Dimitris Plexousakis, Vassilis Christophides, Manolis Koubarakis, Klemens Böhm, and Elena Ferrari, editors, *Advances in Database Technology - EDBT*, pages 183–199, 2004.

[Baldi, 2011] Pierre Baldi. Autoencoders, unsupervised learning and deep architectures. In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27*, UTLW'11, pages 37–50, 2011.

[Baowaly *et al.*, 2018] Mrinal Kanti Baowaly, Chao-Lin Liu, Chia-Ching Lin, and Kuan-Ta Chen. Synthesizing electronic health records using improved generative adversarial networks. *Journal of the American Medical Informatics Association*, 26(3):228–241, 12 2018.

[Chawla *et al.*, 2002] Nitesh Chawla, Kevin Bowyer, Lawrence Hall, and Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[Choi *et al.*, 2017] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun. Generating multi-label discrete electronic health records using generative adversarial networks. *CoRR*, abs/1703.06490, 2017.

[Demidova and Klyueva, 2017] L. Demidova and I. Klyueva. Svm classification: Optimization with the smote algorithm for the class imbalance problem. In *2017 6th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–4, 2017.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems (NIPS)*, pages 2672–2680, 2014.

[Gulrajani *et al.*, 2017] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5767–5777, 2017.

[Han *et al.*, 2005] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In *Proceedings of the International Conference on Advances in Intelligent Computing*, 2005.

[He *et al.*, 2008] Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 1322–1328, 2008.

[Hinton and Salakhutdinov, 2006] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[Isola *et al.*, 2017] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1125–1134, 2017.

[Kohavi, 1996] Ron Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 202–207. AAAI Press, 1996.

[Lemaître *et al.*, 2017] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

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

[Pascual *et al.*, 2017] Santiago Pascual, Antonio Bonafonte, and Joan Serrà. Segan: Speech enhancement generative adversarial network. *arXiv preprint arXiv:1703.09452*, 2017.

[Pele and Werman, 2009] Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *IEEE 12th International Conference on Computer Vision (ICCV)*, pages 460–467, 2009.

[Radford *et al.*, 2015] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[Winters, 1960] Peter R. Winters. Forecasting sales by exponentially weighted moving averages. *Management Science*, 6(3):324–342, 1960.

[Zhu *et al.*, 2017] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2223–2232, 2017.