Conditional Data Synthesis with Deep Generative Models for Imbalanced Dataset Oversampling

Leonidas Akritidis
School of Science and Technology
Int'l Hellenic University
Thessaloniki, Greece
Email: lakritidis@ihu.gr

Athanasios Fevgas
Department of Electrical and Computer Engineering
University of Thessaly
Volos, Greece
Email: fevgas@e-ce.uth.gr

Miltiadis Alamaniotis
Department of Electrical and Computer Engineering
University of Texas at San Antonio
San Antonio, USA
Email: Miltos.Alamaniotis@utsa.edu

Panayiotis Bozanis
School of Science and Technology
Int'l Hellenic University
Thessaloniki, Greece
Email: pbozanis@ihu.gr

***Abstract*—The problem of data imbalance is defined as the uneven distribution of the training examples to the existing classes of a dataset. Among a wide variety of solutions, the oversampling techniques try to mitigate the problem by synthesizing artificial examples associated with the minority class. The huge success of Generative Adversarial Networks (GANs) rendered them an attractive choice for oversampling and numerous researchers proposed modifications of GANs for imbalanced datasets. Nevertheless, the existing models employ the entire minority class for sample generation, thus being vulnerable to outliers and noisy data instances. In addition, the majority of the relevant research concerns image classification tasks, leaving a large gap for research with tabular data. Finally, another powerful and popular generative model, the Variational Autoencoder (VAE) has been rather overlooked by the community in class imbalance solutions. In this paper we present SB-GAN and SB-VAE, two generative models that identify borderline and noisy samples before they are trained. In this manner SB-GAN and SB-VAE learn better class distributions that are not distorted by the existence of outliers. The experimental evaluation of SB-GAN and SB-VAE with 4 tabular datasets revealed a superior performance against 8 state-of-the-art oversampling techniques.**

***Index Terms*—imbalanced datasets, oversampling, generative models, GAN, VAE**

I. Introduction

Class imbalance concerns data collections consisting of samples with an uneven distribution to the underlying classes. An intrusion detection dataset with 100 legitimate actions and 5 malicious attempts is a representative example of class imbalance. Nowadays, imbalanced datasets are present in almost every classification task including intrusion, malware, and fraud detection, object recognition, sentiment analysis, medical diagnosis, and so on.

Training classifiers with imbalanced data degrades their performance significantly, because their output models are strongly biased towards the majority class. The minority class is not learned effectively, so the classifiers do not generalize well. The importance of the problem has attracted the attention of multiple researchers who introduced a wide variety of methods to address it. These methods are usually classified into 4 general categories: cost sensitive, algorithm based, data preprocessing, and hybrid methods [1].

The four aforementioned categories can be further divided into tens of smaller subcategories. In this work we are dealing with one of the most popular among them, the *oversampling* techniques, which fall into the category of data preprocessing methods. Oversampling refers to the generation of synthetic samples, associated with the minority class, with the aim of alleviating the problem of class imbalance. SMOTE and its dozens of variants are among the most traditional oversampling approaches [2], [3].

During the past years, enormous advances have been made to the field of generative modelling. In this context, Generative Adversarial Networks (GANs) are architectures comprised of two parts: the Generator and the Discriminator [4]. Both models are trained simultaneously in an adversarial fashion, where: i) the Generator produces fake samples with the aim of deceiving the Discriminator, and ii) the Discriminator tries to predict whether an example is fake or real. GANs have been primarily utilized to confront imbalance problems in computer vision tasks, leaving much space for research with tabular data. Indicatively, the recent survey of Sampath et al. studied numerous papers using GANs for image generation only [5].

The Variational Autoencoder (VAE) is another popular generative framework based on the Autoencoder architecture [6], [7]. VAEs encode an input sample as a probability distribution over the latent space (instead of encoding it as a simple vector). The encoded distribution, usually a Gaussian, is regularized during training to ensure that the latent space has meaningful properties. Despite their good performance, VAEs have been rather overlooked by the researchers, since the majority of the relevant works utilize GANs for oversampling purposes.

In this paper we introduce two variants of GAN and VAE, termed SB-GAN ans SB-VAE, respectively. Unlike the existing works, both models are fed not with the entire minority class, but only with carefully selected data instances. In short, the selection criterion depends on the classes of the neighboring data points. In particular, in case all the neighbors of a training sample belong to different classes, then the sample is considered as an outlier and as such, it does not

This research is co-financed by Greece and the European Union (European Social Fund-SF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning 2014-2020" in the context of the project "Support for International Actions of the International Hellenic University" (MIS 5154651).

comply with the distribution of its own class. In SB-GAN/SB-VAE such samples are ignored during training, thus improving the learned class distributions.

The contributions of this paper are summarized in the following list:

- We introduce SB-GAN and SB-VAE, two conditional models based on GAN and VAE respectively, for minority data oversampling in imbalanced datasets. Both of them include a preprocessing layer that identifies the outlier and noisy examples and excludes them from training. Therefore, the models learn improved distributions of the minority class from safe and borderline data instances.
- The majority of relevant works utilize GANs to mitigate class imbalance in computer vision tasks. In contrast, this work utilizes GANs to synthesize minority tabular data.
- Despite their proved effectiveness in many data generation applications, VAEs have been rather overlooked in class imbalance tasks. Here we study such models and we show that they can be competitive to GANs.
- We conduct experiments on 4 public imbalanced datasets that demonstrate the effectiveness of SB-GAN and SB-VAE against 8 traditional oversampling methods.

The rest of this paper is organized as follows: In Section II we present a brief overview of the relevant literature in the field of imbalanced data. Section III refers to preliminary elements from the theory of GANs and VAEs, whereas Section IV describes the proposed SB-GAN and SB-VAE models. The usefulness of both models is assessed in the experimental evaluation of Section V. Finally, Section VI summarizes the conclusions of this work and outlines insights for further study.

II. Related Work

The importance of the data imbalance problem has attracted the attention of researchers and led to the introduction of a wide variety of relevant methods. These methods are typically classified into 4 categories: cost sensitive approaches, data preprocessing, algorithm-based techniques, and hybrid solutions.

The methods of the first category adjust the training process of a learner to decrease its bias towards the majority class. For this purpose, a two-dimensional misclassification matrix is formed with the aim of penalizing more heavily the class mispredictions of the minority samples. The literature contains cost sensitive training algorithms for Feed-Forward Neural Networks (FFNNs) [8], [9], Convolutional Neural Networks (CNNs) [10], [11], Decision Trees [12], [13], Support Vector Machines (SVMs) [14], and so on. The most significant challenge in cost sensitive learning is how to set the weights of the misclassification matrix. In many cases, the optimal setting is application-specific and it is performed by human experts.

On the other hand, the data preprocessing techniques attempt to augment the imbalanced data either by *undersampling* the majority class, or by *oversampling* the minority class, or by executing a *hybrid sampling* pipeline that performs a combination of oversampling and undersampling.

The undersampling methods bring balance to a dataset by reducing the population of the majority class samples. This is achieved either through simple sample removal, or through the generation of a small number of representative samples. In Random Undersampling, a number of samples is randomly pruned from the majority class until the desired imbalance ratio is obtained [15]. Alternatively, a clustering algorithm can be applied at the samples of the majority class with the aim of replacing the contents of each cluster by a single representative element. In this spirit, the authors of [16] applied k-Means at the majority class and replaced the contents of each cluster either by the centroid points, or by their nearest neighbor. The method of [17] applied clustering at the entire dataset, to generate a collection of mixed clusters that contain data points from both the majority and the minority classes.

The most common risk in undersampling is the information loss that takes place after the removal of important majority samples. Significant problems may also arise by the distortion of the class's probability distribution [1].

Another way to establish balance in a dataset is to increase the population of the minority samples with an oversampling technique. Random Oversampling (ROS) is the simplest method [18]. In contrast, Adaptive Synthetic Sampling (ADASYN) works by examining the nearest neighbors of a minority sample; the more majority nearest neighbors, the more synthetic examples are created [19].

The Synthetic Minority Oversampling Technique (SMOTE) initially identifies the k nearest neighbors of a minority sample [2]. Then, minority samples are randomly synthesized over the line that connects the sample with each of its neighbors. SMOTE is effective because the synthetic minority examples are relatively close to the existing ones. Therefore, the class distribution is not affected much. On the other hand, the synthetic examples are created without considering the majority class, so there is a possibility of synthesizing ambiguous examples if there is a strong overlap for the classes.

To overcome this problem, a large number of SMOTE variants have been proposed over the past years. Borderline-SMOTE categorizes each minority sample as noise, in-danger, and safe according to the classes of its nearest neighbors [20]. Then, it generates synthetic data by applying SMOTE only at the samples in-danger, i.e. the ones that have at least half their nearest neighbors belonging to the same class. The survey of [3] refers to numerous SMOTE variants with SMOTE-SVM [21] and kMeans-SMOTE [22] being the most significant among them. SMOTE has also been used with ensemble learning models; SMOTEBoost adapts the AdaBoost.M2 algorithm by applying SMOTE before training a weak learner [23]. Similarly, SMOTEBagging generates samples with SMOTE at each training round of a bagging classifier [24].

The excellent generation capabilities of GANs attracted many researchers to suggest GAN-based techniques for mitigating class imbalance. Representative models include the Conditional GAN (cGAN) [25], Auxiliary Classifier GAN (ACGAN) [26], Info-GAN [27], Deep Convolutional GAN (DCGAN) [28], and numerous others. A systematic survey of the most important GAN models for imbalanced datasets in computer vision tasks has been published in [5].

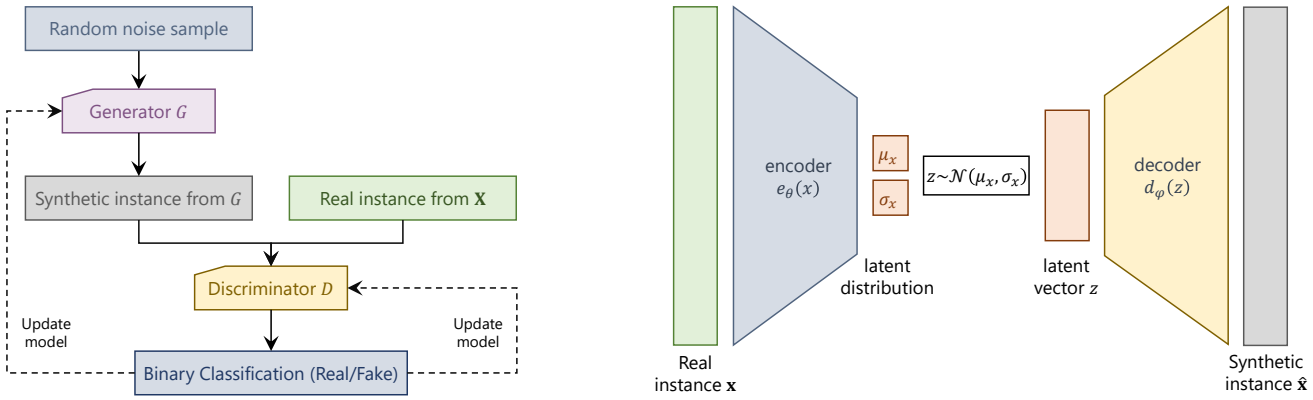

Fig. 1. Architecture and training of a Generative Adversarial Network (left) and Variational Autoencoder (right).

III. Generative Models

This section presents briefly the building blocks and the fundamental elements from the theory of GANs and VAEs.

A. Conditional Generative Adversarial Networks (cGANs)

GANs are among the leading architectures in the area of generative modeling, since they have been proved effective in synthetic data generation [4]. Inside a GAN, two Neural Networks compete each other with the aim of learning the target distribution and generating artificial data: a Discriminator D and a Generator G (left part of Fig. 1).

The Discriminator D is a binary classifier trained to distinguish whether its input data samples are real or not. For this reason, D trained with both real and fake examples. The real examples are drawn from the original training set $\mathbf{X}$, whereas the fake ones are synthesized by the Generator G. The employed loss function must penalize all misclassifications, namely, the cases where real data is identified as fake, or fake data identified as real. At each iteration, D updates its parameters using backpropagation, thus improving its ability to identify fake data instances.

On the other hand, the Generator G learns to synthesize artificial data instances by using the output of D. Its goal is to deceive the discriminator D, so that its output is classified as real. The training process of G takes place simultaneously with that of D and involves the following phases: Initially, a random noise sample is used it to produce the output of G. The output of G is fed to D and the discriminator loss is computed. In the sequel, the error is backpropagated to compute the necessary gradients and update only the weights of G.

Formally, D and G play the following zero-sum game:

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

In many cases, the objective is not to simply generate artificial data, but to synthesize data instances belonging to a particular class. Conditional GANs (CGANs) address this requirement by receiving as inputs both the samples and their respective classes $\mathbf{y}$ [25]. This applies to both Discriminator and Generator, so the aforementioned zero-sum game becomes:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_\mathbf{x}(\mathbf{x}|)}[\log(D(\mathbf{x}|\mathbf{y}))] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{y})))] \quad (2)$$

During CGAN training, the one-hot-encoded vectors $\mathbf{y}$ are fed to both the Discriminator and the Generator, after they have been concatenated with the input feature vectors (either fake, or real ones) [25].

B. Conditional Variational Autoencoders (cVAEs)

Autoencoders are neural networks that implement the Encoder E-Decoder D architecture (right part of Fig. 1). They are a form of an unsupervised learning model, trained to reproduce its input $\mathbf{x}$ as accurately as possible. For this purpose, E encodes $\mathbf{x}$ producing a low-dimensional latent representation $\mathbf{z} = e(\mathbf{x})$, whereas D decodes $\mathbf{z}$ outputting a vector $\hat{\mathbf{x}} = d(\mathbf{z}) = d(e(\mathbf{x}))$. If $\hat{\mathbf{x}}$ is the output of the network, then the Autoencoder minimizes the reconstruction error $\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}})$ that quantifies the distance between $\mathbf{x}$ and $\hat{\mathbf{x}}$.

The Autoencoder has been proved successful in dimensionality reduction tasks, but it cannot be used for data generation. The reason is that the Encoder cannot regularize the latent space $\mathbf{Z}$ appropriately. It is difficult to predict the distribution of values in that space, therefore, taking random samples from it and feeding them to the decoder will most likely produce meaningless instances. Finding a latent value $\mathbf{z} \in \mathbf{Z}$ for which the decoder will produce meaningful data is almost impossible.

VAE is an Autoencoder whose encoded distribution is regularized during training to ensure that the latent space has suitable properties for generating useful instances [6]. In this context, the input $\mathbf{x}$ of a VAE is not encoded as a simple latent vector $\mathbf{z}$, but as a probability distribution $p_\theta(\mathbf{x}|\mathbf{z})$ over the latent space $\mathbf{Z}$. Then, useful data instances are generated by D by feeding it with samples drawn randomly from $p_\theta(\mathbf{x}|\mathbf{z})$.

In practice, the encoded distribution is $p_\theta(\mathbf{x}|\mathbf{z})$ chosen to be normal with a mean value $\mu_\mathbf{x} = 0$ and a standard deviation $\sigma_\mathbf{x} = 1$. The loss function is a linear combination of the recon-

struction loss and the Kulback-Leibler divergence between the returned distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and a standard Gaussian, namely:

$$L_{\text{VAE}} = -KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] \quad (3)$$

Similarly to CGANs, Conditional VAEs introduce the one-hot-encoded class vector $\mathbf{y}$ to the training process to enable the generation of class-specific data instances [29]. Since $\mathbf{y}$ is generated from the distribution $p_\theta(\mathbf{y}|\mathbf{x},\mathbf{z})$ and $\mathbf{z}$ is drawn from the prior $p_\theta(\mathbf{z}|\mathbf{x})$, the loss function of a CVAE is:

$$L_{\text{CVAE}} = -KL(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})||p_\theta(\mathbf{z}|\mathbf{x})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log p_\theta(\mathbf{y}|\mathbf{x},\mathbf{z})]$$

IV. Safe-Borderline Sample Selection

The enormous success of the recent generative models primarily comes from their ability to learn the probability distribution of the data on which they are trained. On the other hand, the outliers are observations that lie in an abnormal distance from the other samples of a population. In other words, the outliers usually do not comply with the probability distribution of their population.

Apparently, including outliers in the generator training process reduces its ability to effectively learn the underlying distribution. In this paper we propose a method for providing a generative model with the appropriate data that will enable it learn better distributions of the minority class.

Let $\mathbf{X}$ be the matrix that contains all training vectors. Now we define $\mathbf{X}^{(i)}$ and $\mathbf{X}^{(a)}$ as the matrices that contain the training examples belonging to the minority and majority class, respectively.

The *Safe-Borderline (SB)* technique specifies a hyper-sphere $\mathcal{S}$ of radius r around each minority sample $\mathbf{x}_j^{(i)} \in \mathbf{X}^{(i)}$. Then, $\mathbf{x}_j^{(i)}$ is tagged according to the classes of the elements that lie inside its surrounding hyper-sphere $\mathcal{S}$, as follows:

- *Outlier*: If all samples inside $\mathcal{S}$ belong to a different class than $\mathbf{x}_j^{(i)}$.
- *Isolated*: If $\mathcal{S}$ is empty.
- *Borderline*: If $\mathcal{S}$ contains mixed samples from all classes.
- *Safe, or Core*: If all the samples inside $\mathcal{S}$ belong to the same class as $\mathbf{x}_j^{(i)}$.

From these five groups the appropriate samples will be selected for training SB-GAN and SB-VAE. The entire process is shown in Algorithm 1.

Algorithm 1 My algorithm

Input: Training set $\mathbf{X}$, radius r, α
Output: Safe-Borderline set $\mathbf{X}_{SB}$

1: $\mathbf{X}^{(i)} \leftarrow [\,]$ *// minority class examples*
2: **for** each training example $\mathbf{x}_j \in \mathbf{X}$ **do**
3: **if** $y_j == y^{(i)}$ **then** $\mathbf{X}^{(i)} \leftarrow \mathbf{X}^{(i)}.append(\mathbf{x})$
4:
5: $\mathbf{X}_{iso}^{(i)} \leftarrow [\,]$ *// isolated samples*
6: $\mathbf{X}_{out}^{(i)} \leftarrow [\,]$ *// outliers*
7: $\mathbf{X}_{sf}^{(i)} \leftarrow [\,]$ *// safe samples*
8: $\mathbf{X}_{b}^{(i)} \leftarrow [\,]$ *// borderline samples*
9: **for** each minority sample $\mathbf{x}_j \in \mathbf{X}^{(i)}$ **do**
10: $\mathbf{X}_{j,nn} \leftarrow$ FetchPointsInRadius$(\mathbf{X}, \mathbf{x}_j, r)$
11: **if** $|\mathbf{X}_{j,nn}| == 0$ **then**
12: $\mathbf{X}_{iso}^{(i)} \leftarrow \mathbf{X}_{iso}^{(i)}.append(\mathbf{x}_j)$
13: **break**
14: $outlier \leftarrow$ True
15: $safe \leftarrow$ True
16: **for** each sample $\mathbf{x}_k \in \mathbf{X}_{j,nn}$ inside radius of $\mathbf{x}_j$ **do**
17: **if** $y_k == y_j$ **then**
18: $outlier \leftarrow$ False
19: **else**
20: $safe \leftarrow$ False
21: **if** $outlier ==$ True **then** $\mathbf{X}_{out}^{(i)} \leftarrow \mathbf{X}_{out}^{(i)}.append(\mathbf{x}_j)$
22: **else if** $safe ==$ True **then** $\mathbf{X}_{sf}^{(i)} \leftarrow \mathbf{X}_{sf}^{(i)}.append(\mathbf{x}_j)$
23: **else** $\mathbf{X}_{b}^{(i)} \leftarrow \mathbf{X}_{b}^{(i)}.append(\mathbf{x}_j)$
24:
25: $\mathbf{X}_{SB} \leftarrow \mathbf{X}_{sf}^{(i)}$
26: **if** $|\mathbf{X}_{SB}| < \alpha|\mathbf{X}^{(i)}|$ **then** $\mathbf{X}_{SB} \leftarrow \mathbf{X}_{SB}.extend(\mathbf{X}_{b}^{(i)})$
27: **if** $|\mathbf{X}_{SB}| < \alpha|\mathbf{X}^{(i)}|$ **then** $\mathbf{X}_{SB} \leftarrow \mathbf{X}_{SB}.extend(\mathbf{X}_{iso}^{(i)})$
28: **return** $\mathbf{X}_{SB}$

The algorithm is logically divided in 3 stages. The first stage constructs the matrix that contains the minority training examples $\mathbf{X}^{(i)}$ (steps 1–3). The second stage begins by initializing 4 matrices: $\mathbf{X}_{out}^{(i)}$, $\mathbf{X}_{iso}^{(i)}$, $\mathbf{X}_{b}^{(i)}$, and $\mathbf{X}_{sf}^{(i)}$ that will accommodate the outliers, the isolated samples, the borderline samples, and the safe points respectively (steps 5–8).

In the sequel, for each sample belonging to the minority class $\mathbf{x}_j \in \mathbf{X}^{(i)}$, the range query FetchPointsInRadius is executed to retrieve all the neighbors the lie into distances smaller than r from $\mathbf{x}_j$ (step 10). If no neighbors exist, then $\mathbf{x}_j$ is immediately marked as isolated; $\mathbf{X}_{iso}^{(i)}$ is updated and the process continues to the next sample (steps 11–13). Otherwise, the classes of the nearest neighbors are subsequently examined and according to the aforementioned rules, $\mathbf{x}_j$ is classified as an outlier, borderline sample, or safe sample. The stage ends by updating the corresponding matrices (steps 21–23).

Eventually, stage 3 constructs the final training set $\mathbf{X}_{SB}$ that will be fed to the generative model (steps 25–27). All safe points of $\mathbf{X}_{sf}^{(i)}$ are immediately appended to $\mathbf{X}_{SB}$. However, the minority class may consist of only a few scattered samples, so $\mathbf{X}_{SB}$ may be considerably smaller than the original training set. To avoid discarding too many valuable samples, we use a parameter $\alpha \in [0, 1]$ (here we set $\alpha = 0.7$) to include additional borderline, or isolated instances to $\mathbf{X}_{SB}$.

We close this section with some remarks on the function FetchPointsInRadius. Finding all the neighboring points within a hyper-sphere of radius r is an expensive operation, due to the involved quadratic time complexity. The process can be accelerated by using an auxiliary spatial data structure like a KDTree, or a Ball Tree, however, if the data is highly dimensional, a brute force approach may be more efficient. Notice that this is a drawback of all methods using nearest neighbor approaches, including SMOTE and its variants.

TABLE I
BENCHMARK DATASETS

Name	Samples	Dimensions	IR
Rice Type	18185	10	1:1.2
Smoke Detection	62630	14	1:2.5
Credit Card Default	30000	25	1:3.5
Surgical	14635	24	1:3.0

V. EXPERIMENTS

This section demonstrates the usefulness of SB-GAN and SB-VAE in the task of minority class oversampling in imbalanced datasets. Their effectiveness is compared against a collection of other well-established oversampling techniques and their performance is highlighted and discussed.

All the experiments that we present here have been carried out on a typical Windows 10 workstation with an Intel Core I7 12700K CPU, 32 GB RAM and an NVIDIA RTX 3060 GPU with 12 GB of video RAM.

A. Datasets

The experiments have been conducted by employing four publicly accessible datasets: Rice Type Classification[1], Smoke Detection[2], Credit Card Default[3] and Surgical Dataset[4]. All have include 2 classes and were selected because they possess three desired properties: i) they are imbalanced, ii) they contain an explicit class column that makes them suitable for classification, and iii) they include only numerical features that simplifies their processing. Their attributes are shown in Table I; the fourth column denotes the imbalance ratio.

The first dataset contains 18185 examples of two rice classes and it is the most balanced one (imbalance ratio 1:1.2). Smoke Detection is the largest dataset, comprising 62630 cases of activation (or not) of a photoelectric smoke detector. The 14 attributes of each vector represent a reading from a different sensor. The third dataset is the most imbalanced one (imbalance ratio 1:3.5) and contains 30000 cases of customer default payments in Taiwan. Finally, the Surgical dataset concerns the outcome of 14635 surgical operations according to 24 features.

B. SB-GAN/SB-VAE Architectures & Adversary Oversamplers

This section describes several architectural details of SB-GAN and SB-VAE. The Discriminator of SB-GAN comprised two dense (fully connected) layers with 64 and 32 neurons respectively. Each dense layer was followed by a Dropout layer (factor equal to 0.5) to prevent the model from overfitting. Leaky ReLU was the selected function for neuron activation. On the other hand, the Generator included two residual layers and one dense layer. The residual blocks comprised 64 and 32 neurons respectively, followed by a batch normalization layer.

Regarding SB-VAE, the Encoder included two dense layers with 64 and 32 neurons respectively; each layer was followed by a batch normalization layer. The architecture of the Decoder was identical, but mirrored. The dimensionality of the latent vectors μ and σ and the sampling vector $\mathbf{z}$ was set equal to 8.

Eight oversampling techniques and models have been used for comparison: Random Oversampling (ROS) [18], Synthetic Minority Oversampling Technique (SMOTE) [2], Borderline SMOTE (BRD-SMOTE) [20], SVM-SMOTE [21], k-Means SMOTE (KMN-SMOTE) [22], Adaptive Synthetic Sampling (ADASYN) [19], a Generative Adversarial Network (GAN) [4], and a Variational Autoencoder (VAE) [6]. The architectures of Generator/Discriminator and Encoder/Decoder of the last two generative models were identical to those of SB-GAN and SB-VAE, respectively.

C. Classifiers: Tuning, Training, Evaluation

The effectiveness of SB-GAN and SB-VAE has been measured in the context of how much they improve the performance of various classification models. In particular, the following 4 classifiers have been utilized:

- Feed-Forward Neural Network (FFNN): We implemented a typical fully-connected feed-forward neural network with two hidden layers including 16 and 4 neurons, respectively. Rectified Linear Unit (ReLU) was selected as the activation function for the hidden layers. The network was trained with the Adam optimizer that minimized the binary cross-entropy loss function.
- Support Vector Machines (SVM): The Radial Basis Function (RBF) with L2 regularization was utilized for fitting. We set the regularization parameter equal to $C = 1.0$.
- Random Forest (RF): A forest with 50 weak estimators was used. The estimators were standard Decision Trees with no restrictions to their maximum depth and leaf size.
- Logistic Regression (LR): The Limited-memory BFGS algorithm with L2 regularization was employed for learning the parameters of the logistic function.

The only feature processing technique that we applied was a simple transformation of their values via standardization. The models were evaluated by using the popular 5-fold cross validation method that dictated a 80%/20% split ratio for the training and test set. During each fold, and before each model was trained, the training set was firstly standardized and then oversampled. Thus, at each cross validation stage, a pipeline of the form *(Standardize, Oversample, Train)* was serially executed. Then, the values of the desired evaluation measures were recorded by averaging the 5 values returned by each fold.

Regarding the evaluation metrics, $Accuracy$ is not suitable when dealing with imbalanced datasets, because it erroneously receives high values that hide the inability of a classifier to predict the minority class. A dummy classifier that always predicts 0 would achieve $Accuracy = 0.95$ on a binary dataset with 100 test samples, of which 95 belong to class 0 and 5 to class 1. Clearly, $Accuracy$ cannot capture the inability of this dummy classifier to predict class 1.

In this work we adopt two popular evaluation measures, $F1$ and AUC (Area Under the Receiver Operating Curve). Both of them are based on the True/False Positive/Negative values that

[1]https://bit.ly/46YFYg4
[2]https://bit.ly/3Opi5a1
[3]bit.ly/3rDwfLY
[4]https://bit.ly/473zEUH

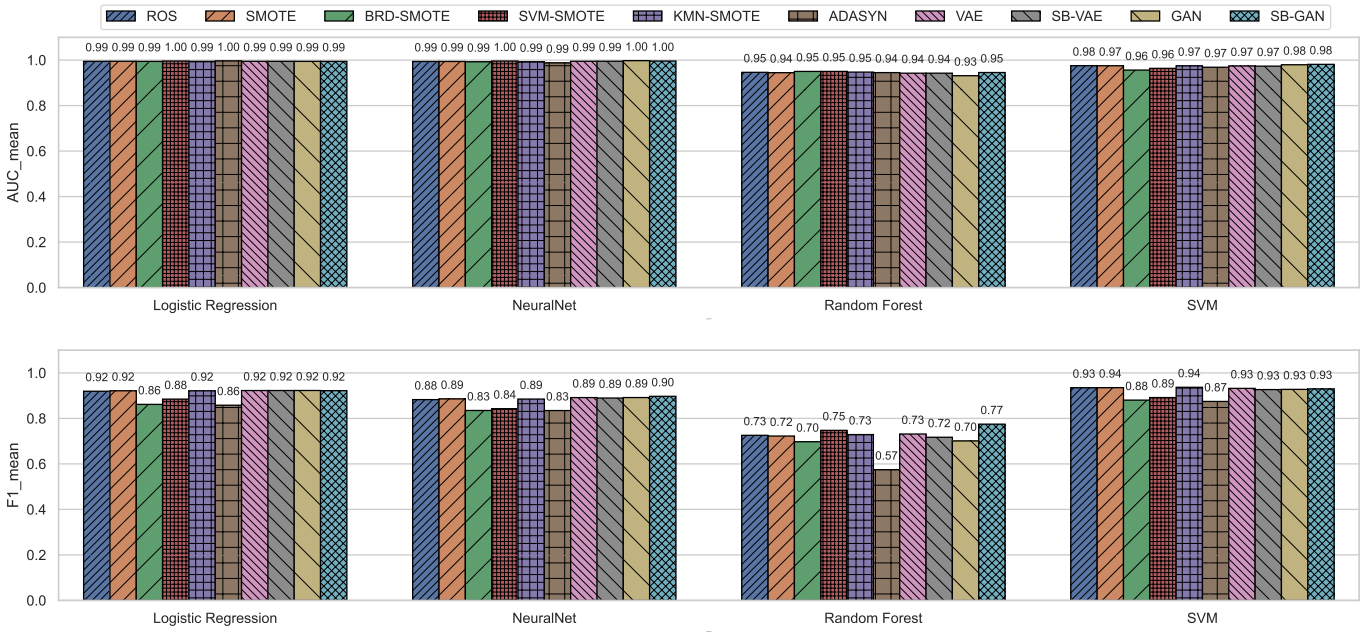

Fig. 2. Performance comparison for different classifiers and oversampling techniques for the Rice Type Classification dataset.

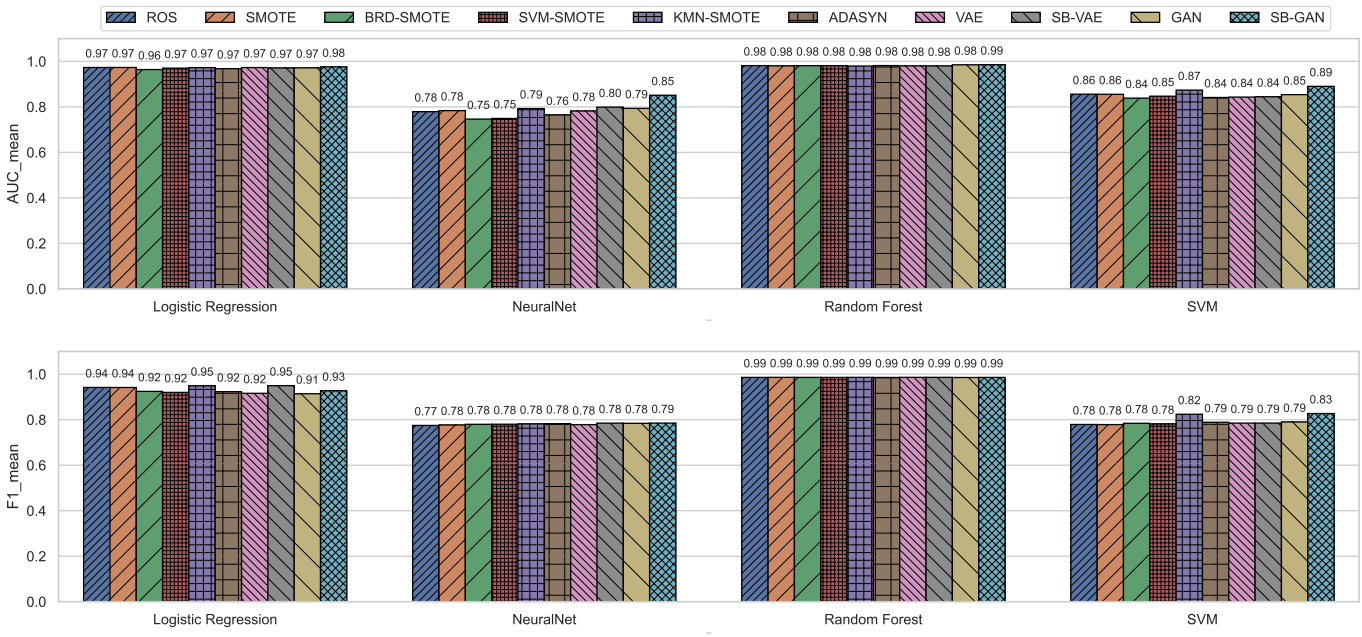

Fig. 3. Performance comparison for different classifiers and oversampling techniques for the Smoke Detection dataset.

respectively represent the number of the correct and incorrect predictions of a model on the positive and negative classes. These values enable the introduction of additional measures:

$$\begin{aligned} \text{Precision} &= TP/(TP+FP) \\ \text{Recall} = \text{Sensitivity} &= TPR = TP/(TP+FN) \\ \text{Specificity} &= TNR = TN/(TN+FP) \\ F1 &= 2 \cdot \text{Precision} \cdot \text{Recall}/(\text{Precision} + \text{Recall}) \end{aligned}$$

The Receiver Operating Curve (ROC) plots TPR vs. FPR at different classification thresholds. Area Under the Curve (AUC) is a measure of the surface of the region below the entire ROC curve.

D. Results & Discussion

This section presents the results of the performance evaluation of SB-GAN and SB-VAE. Figures 2, 3, 4, and 5 depict the values of AUC (upper diagram) and $F1$ (lower diagram) for the Rice Type Classification, Smoke Detection, Credit Card Default, and Surgical datasets, respectively.

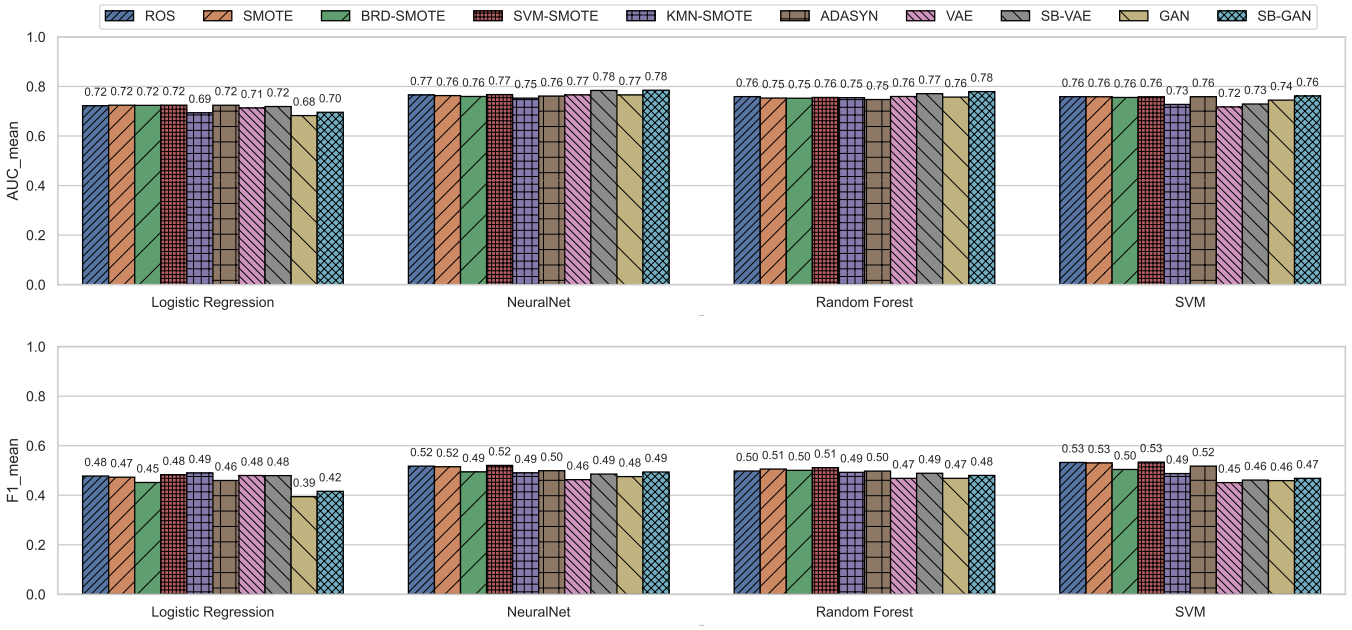

Fig. 4. Performance comparison for different classifiers and oversampling techniques for the Credit Card Default dataset.

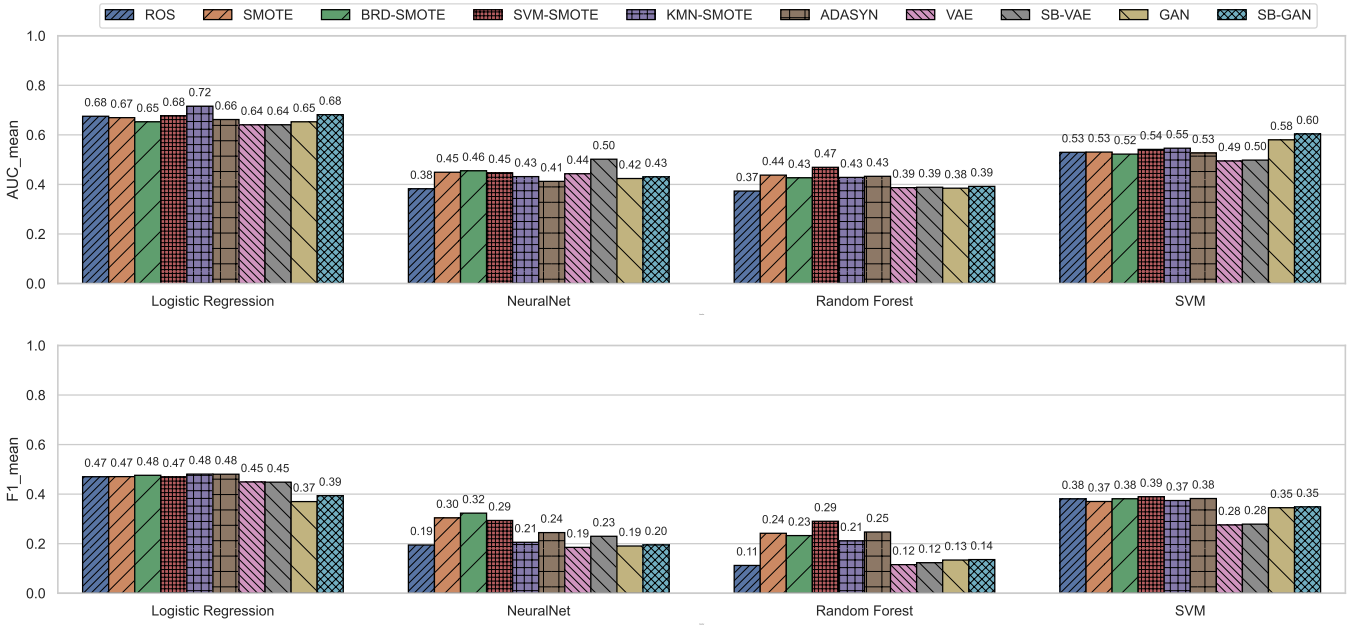

Fig. 5. Performance comparison for different classifiers and oversampling techniques for the Surgical dataset.

Regarding the Rice Type Classification dataset (Fig. 2), LR, FFNN, and SVM achieved very high values for AUC, regardless of the applied oversampling technique. This is was anticipated to some extent, since this dataset has a rather small imbalance ratio (1:1.2). On the other hand, in terms of $F1$, there were some significant discrepancies. The best results were obtained by using SMOTE and k-Means SMOTE in combination with SVM ($F1 \simeq 0.94$); all generative models also had a satisfactory performance ($F1 \simeq 0.94$). The best $F1$ measurements of the FFNN and RF classifiers were achieved by oversampling the minority class with SB-GAN.

In contrast to the Rice Type Classification dataset, RF was the best classifier in the Smoke Detection Dataset. All oversampling methods achieved almost equal AUC and $F1$ values (between 0.98 and 0.99), and SB-GAN again outperformed the other methods by a slight margin. FFNN and SVM were now the weakest classifiers, but they both maximized their effectiveness with SB-GAN (in terms of both AUC and $F1$).

FFNN was the winning classifier in the Credit Card Default dataset (Fig. 4). Similarly to the two previous datasets, FFNN

maximized its achieved AUC in combination with SB-GAN ($AUC \simeq 0.78$). The same applies to SVM and RF.

The only case where SB-GAN was not the best method, was observed in the Surgical dataset. The best classifier was LR and maximized its AUC and $F1$ with k-Means SMOTE.

A more general observation of these results produces the following conclusions:

- In each of the 4 examined datasets, we had a different classification model as a winner. However, in 3 out of 4 cases, the winners maximized their performance in combination with SB-GAN.
- In almost all cases, SB-GAN outperformed the typical GAN model by 1% to about 8%. Similarly, SB-VAE defeated VAE by margins ranging between 1% and 6.5%.
- Regardless of the applied oversampling technique, the performance of all classifiers drops as the imbalance ratio of a dataset increases.
- Although slightly worse than SB-GAN, the older oversampling techniques are still very effective.

These conclusions verify the significance of the imbalance problem and highlight the usefulness of the proposed models.

VI. Conclusions and Future Work

In this paper we presented SB-GAN and SB-VAE, two conditional generative models for oversampling in imbalanced datasets. Both models adopt an approach that filters out the input training examples, effectively removing outliers and other noisy samples. By feeding filtered data to SB-GAN and SB-VAE, the models can better approximate the distribution of the minority class and output more meaningful samples. The experimental evaluation of SB-GAN and SB-VAE on four public imbalanced datasets yielded promising results, since in most cases, they outperformed the baseline generative models.

Future work includes a deeper experimentation with the training process of SB-GAN. Instead of feeding the Generator with random noise, we shall study the potential of feeding it with samples that respect the class imbalance ratio of the input data. In this way we anticipate to create robust Generators that can better compete with the model's Discriminator.

References

[1] H. Kaur, H. S. Pannu, and A. K. Malhi, "A systematic review on imbalanced data challenges in machine learning: Applications and solutions," *ACM Computing Surveys*, vol. 52, no. 4, pp. 1–36, 2019.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[3] A. Fernández, S. Garcia, F. Herrera, and N. V. Chawla, "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary," *Journal of AI Research*, vol. 61, pp. 863–905, 2018.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[5] V. Sampath, I. Maurtua, J. J. Aguilar Martin, and A. Gutierrez, "A survey on Generative Adversarial Networks for imbalance problems in computer vision tasks," *Journal of Big Data*, vol. 8, pp. 1–59, 2021.

[6] D. P. Kingma and M. Welling, "Auto-encoding Variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[7] D. P. Kingma, M. Welling *et al.*, "An introduction to Variational Autoencoders," *Foundations and Trends in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[8] Z.-H. Zhou and X.-Y. Liu, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2005.

[9] C. L. Castro and A. P. Braga, "Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 888–899, 2013.

[10] S. Soleymanpour, H. Sadr, and M. Nazari Soleimandarabi, "CSCNN: cost-sensitive Convolutional Neural Network for encrypted traffic classification," *Neural Processing Letters*, vol. 53, no. 5, pp. 3497–3523, 2021.

[11] H. Zhang, L. Jiang, and C. Li, "CS-ResNet: Cost-sensitive Residual Convolutional Neural Network for PCB cosmetic defect detection," *Expert Systems with Applications*, vol. 185, p. 115673, 2021.

[12] B. Krawczyk, M. Woźniak, and G. Schaefer, "Cost-sensitive decision tree ensembles for effective imbalanced classification," *Applied Soft Computing*, vol. 14, pp. 554–562, 2014.

[13] A. C. Bahnsen, D. Aouada, and B. Ottersten, "Example-dependent cost-sensitive decision trees," *Expert Systems with Applications*, vol. 42, no. 19, pp. 6609–6619, 2015.

[14] R. Obiedat, R. Qaddoura, A.-Z. Ala'M, L. Al-Qaisi, O. Harfoushi, M. Alrefai, and H. Faris, "Sentiment analysis of customers' reviews using a hybrid evolutionary SVM-based approach in an imbalanced data distribution," *IEEE Access*, vol. 10, pp. 22 260–22 273, 2022.

[15] T. Hasanin and T. Khoshgoftaar, "The effects of random undersampling with simulated class imbalance for big data," in *Proceedings of the 2018 IEEE International Conference on Information Reuse and Integration*, 2018, pp. 70–79.

[16] W.-C. Lin, C.-F. Tsai, Y.-H. Hu, and J.-S. Jhang, "Clustering-based undersampling in class-imbalanced data," *Information Sciences*, vol. 409, pp. 17–26, 2017.

[17] S.-J. Yen and Y.-S. Lee, "Cluster-based under-sampling approaches for imbalanced data distributions," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5718–5727, 2009.

[18] A. Moreo, A. Esuli, and F. Sebastiani, "Distributional random oversampling for imbalanced text classification," in *Proceedings of the 39th ACM SIGIR International Conference on Research and Development in Information Retrieval*, 2016, pp. 805–808.

[19] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive Synthetic Sampling approach for imbalanced learning," in *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 1322–1328.

[20] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Proceedings of 2005 International Conference on Intelligent Computing (Advances in Intelligent Computing)*, 2005, pp. 878–887.

[21] H. M. Nguyen, E. W. Cooper, and K. Kamei, "Borderline over-sampling for imbalanced data classification," *International Journal of Knowledge Engineering and Soft Data Paradigms*, vol. 3, no. 1, pp. 4–21, 2011.

[22] G. Douzas, F. Bacao, and F. Last, "Improving imbalanced learning through a heuristic oversampling method based on k-Means and SMOTE," *Information Sciences*, vol. 465, pp. 1–20, 2018.

[23] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," in *Proceedings of the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2003, pp. 107–119.

[24] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Data Mining*, 2009, pp. 324–331.

[25] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," *arXiv preprint arXiv:1411.1784*, 2014.

[26] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 2642–2651.

[27] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing Generative Adversarial Nets," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[28] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with Deep Convolutional Generative Adversarial Networks," *arXiv preprint arXiv:1511.06434*, 2015.

[29] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in Neural Information Processing Systems*, vol. 28, 2015.